Internet Engineering Task Force (IETF)
Request for Comments: 6550
Category: Standards Track
ISSN: 2070-1721

T. Winter, Ed.
P. Thubert, Ed.
Cisco Systems
A. Brandt
Sigma Designs
J. Hui
Arch Rock Corporation
R. Kelsey
Ember Corporation
P. Levis
Stanford University
K. Pister
Dust Networks
R. Struik
Struik Security Consultancy
JP. Vasseur
Cisco Systems
R. Alexander
Cooper Power Systems

March 2012

RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks

Abstract

Low-Power and Lossy Networks (LLNs) are a class of network in which both the routers and their interconnect are constrained. LLN routers typically operate with constraints on processing power, memory, and energy (battery power). Their interconnects are characterized by high loss rates, low data rates, and instability. LLNs are comprised of anything from a few dozen to thousands of routers. Supported traffic flows include point-to-point (between devices inside the LLN), point-to-multipoint (from a central control point to a subset of devices inside the LLN), and multipoint-to-point (from devices inside the LLN towards a central control point). This document specifies the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), which provides a mechanism whereby multipoint-to-point traffic from devices inside the LLN towards a central control point as well as point-to-multipoint traffic from the central control point to the devices inside the LLN are supported. Support for point-to-point traffic is also available.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6550.

Copyright Notice

Table of Contents

1. Introduction

Low-power and Lossy Networks (LLNs) consist largely of constrained nodes (with limited processing power, memory, and sometimes energy when they are battery operated or energy scavenging). These routers are interconnected by lossy links, typically supporting only low data rates, that are usually unstable with relatively low packet delivery rates. Another characteristic of such networks is that the traffic patterns are not simply point-to-point, but in many cases point-to-multipoint or multipoint-to-point. Furthermore, such networks may potentially comprise up to thousands of nodes. These characteristics offer unique challenges to a routing solution: the IETF ROLL working group has defined application-specific routing requirements for a Low-power and Lossy Network (LLN) routing protocol, specified in [RFC5867], [RFC5826], [RFC5673], and [RFC5548].

This document specifies the IPv6 Routing Protocol for LLNs (RPL). Note that although RPL was specified according to the requirements set forth in the aforementioned requirement documents, its use is in no way limited to these applications.

1.1. Design Principles

RPL was designed with the objective to meet the requirements spelled out in [RFC5867], [RFC5826], [RFC5673], and [RFC5548].

A network may run multiple instances of RPL concurrently. Each such instance may serve different and potentially antagonistic constraints or performance criteria. This document defines how a single instance operates.

In order to be useful in a wide range of LLN application domains, RPL separates packet processing and forwarding from the routing optimization objective. Examples of such objectives include minimizing energy, minimizing latency, or satisfying constraints. This document describes the mode of operation of RPL. Other companion documents specify routing Objective Functions. A RPL implementation, in support of a particular LLN application, will include the necessary Objective Function(s) as required by the application.

RPL operations require bidirectional links. In some LLN scenarios, those links may exhibit asymmetric properties. It is required that the reachability of a router be verified before the router can be used as a parent. RPL expects an external mechanism to be triggered during the parent selection phase in order to verify link properties and neighbor reachability. Neighbor Unreachability Detection (NUD) is such a mechanism, but alternates are possible, including

Bidirectional Forwarding Detection (BFD) [RFC5881] and hints from lower layers via Layer 2 (L2) triggers like [RFC5184]. In a general fashion, a detection mechanism that is reactive to traffic is favored in order to minimize the cost of monitoring links that are not being used.

RPL also expects an external mechanism to access and transport some control information, referred to as the "RPL Packet Information", in data packets. The RPL Packet Information is defined in Section 11.2 and enables the association of a data packet with a RPL Instance and the validation of RPL routing states. The RPL option [RFC6553] is an example of such mechanism. The mechanism is required for all packets except when strict source routing is used (that is for packets going Downward in Non-Storing mode as detailed further in Section 9), which by nature prevents endless loops and alleviates the need for the RPL Packet Information. Future companion specifications may propose alternate ways to carry the RPL Packet Information in the IPv6 packets and may extend the RPL Packet Information to support additional features.

RPL provides a mechanism to disseminate information over the dynamically formed network topology. This dissemination enables minimal configuration in the nodes, allowing nodes to operate mostly autonomously. This mechanism uses Trickle [RFC6206] to optimize the dissemination as described in Section 8.3.

In some applications, RPL assembles topologies of routers that own independent prefixes. Those prefixes may or may not be aggregatable depending on the origin of the routers. A prefix that is owned by a router is advertised as on-link.

RPL also introduces the capability to bind a subnet together with a common prefix and to route within that subnet. A source can inject information about the subnet to be disseminated by RPL, and that source is authoritative for that subnet. Because many LLN links have non-transitive properties, a common prefix that RPL disseminates over the subnet must not be advertised as on-link.

In particular, RPL may disseminate IPv6 Neighbor Discovery (ND) information such as the [RFC4861] Prefix Information Option (PIO) and the [RFC4191] Route Information Option (RIO). ND information that is disseminated by RPL conserves all its original semantics for router to host, with limited extensions for router to router, though it is not to be confused with routing advertisements and it is never to be directly redistributed in another routing protocol. A RPL node often combines host and router behaviors. As a host, it will process the options as specified in [RFC4191], [RFC4861], [RFC4862], and [RFC6275]. As a router, the RPL node may advertise the information

from the options as required for the specific link, for instance, in an ND Router Advertisement (RA) message, though the exact operation is out of scope.

A set of companion documents to this specification will provide further guidance in the form of applicability statements specifying a set of operating points appropriate to the Building Automation, Home Automation, Industrial, and Urban application scenarios.

1.2. Expectations of Link-Layer Type

In compliance with the layered architecture of IP, RPL does not rely on any particular features of a specific link-layer technology. RPL is designed to be able to operate over a variety of different link layers, including ones that are constrained, potentially lossy, or typically utilized in conjunction with highly constrained host or router devices, such as but not limited to, low-power wireless or PLC (Power Line Communication) technologies.

Implementers may find [RFC3819] a useful reference when designing a link-layer interface between RPL and a particular link-layer technology.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Additionally, this document uses terminology from [ROLL-TERMS], and introduces the following terminology:

DAG: Directed Acyclic Graph. A directed graph having the property that all edges are oriented in such a way that no cycles exist. All edges are contained in paths oriented toward and terminating at one or more root nodes.

DAG root: A DAG root is a node within the DAG that has no outgoing edge. Because the graph is acyclic, by definition, all DAGs must have at least one DAG root and all paths terminate at a DAG root.

Destination-Oriented DAG (DODAG): A DAG rooted at a single destination, i.e., at a single DAG root (the DODAG root) with no outgoing edges.

DODAG root: A DODAG root is the DAG root of a DODAG. The DODAG root may act as a border router for the DODAG; in particular, it may aggregate routes in the DODAG and may redistribute DODAG routes into other routing protocols.

Virtual DODAG root: A Virtual DODAG root is the result of two or more RPL routers, for instance, 6LoWPAN Border Routers (6LBRs), coordinating to synchronize DODAG state and act in concert as if they are a single DODAG root (with multiple interfaces), with respect to the LLN. The coordination most likely occurs between powered devices over a reliable transit link, and the details of that scheme are out of scope for this specification (to be defined in future companion specifications).

Up: Up refers to the direction from leaf nodes towards DODAG roots, following DODAG edges. This follows the common terminology used in graphs and depth-first-search, where vertices further from the root are "deeper" or "down" and vertices closer to the root are "shallower" or "up".

Down: Down refers to the direction from DODAG roots towards leaf nodes, in the reverse direction of DODAG edges. This follows the common terminology used in graphs and depth-first-search, where vertices further from the root are "deeper" or "down" and vertices closer to the root are "shallower" or "up".

Rank: A node's Rank defines the node's individual position relative to other nodes with respect to a DODAG root. Rank strictly increases in the Down direction and strictly decreases in the Up direction. The exact way Rank is computed depends on the DAG's Objective Function (OF). The Rank may analogously track a simple topological distance, may be calculated as a function of link metrics, and may consider other properties such as constraints.

Objective Function (OF): An OF defines how routing metrics, optimization objectives, and related functions are used to compute Rank. Furthermore, the OF dictates how parents in the DODAG are selected and, thus, the DODAG formation.

Objective Code Point (OCP): An OCP is an identifier that indicates which Objective Function the DODAG uses.

RPLInstanceID: A RPLInstanceID is a unique identifier within a network. DODAGs with the same RPLInstanceID share the same Objective Function.

RPL Instance: A RPL Instance is a set of one or more DODAGs that share a RPLInstanceID. At most, a RPL node can belong to one DODAG in a RPL Instance. Each RPL Instance operates independently of other RPL Instances. This document describes operation within a single RPL Instance.

DODAGID: A DODAGID is the identifier of a DODAG root. The DODAGID is unique within the scope of a RPL Instance in the LLN. The tuple (RPLInstanceID, DODAGID) uniquely identifies a DODAG.

DODAG Version: A DODAG Version is a specific iteration ("Version") of a DODAG with a given DODAGID.

DODAGVersionNumber: A DODAGVersionNumber is a sequential counter that is incremented by the root to form a new Version of a DODAG. A DODAG Version is identified uniquely by the (RPLInstanceID, DODAGID, DODAGVersionNumber) tuple.

Goal: The Goal is an application-specific goal that is defined outside the scope of RPL. Any node that roots a DODAG will need to know about this Goal to decide whether or not the Goal can be satisfied. A typical Goal is to construct the DODAG according to a specific Objective Function and to keep connectivity to a set of hosts (e.g., to use an Objective Function that minimizes a metric and is connected to a specific database host to store the collected data).

Grounded: A DODAG is grounded when the DODAG root can satisfy the Goal.

Floating: A DODAG is floating if it is not grounded. A floating DODAG is not expected to have the properties required to satisfy the goal. It may, however, provide connectivity to other nodes within the DODAG.

DODAG parent: A parent of a node within a DODAG is one of the immediate successors of the node on a path towards the DODAG root. A DODAG parent's Rank is lower than the node's. (See Section 3.5.1).

Sub-DODAG: The sub-DODAG of a node is the set of other nodes whose paths to the DODAG root pass through that node. Nodes in the sub-DODAG of a node have a greater Rank than that node. (See Section 3.5.1).

Local DODAG: Local DODAGs contain one and only one root node, and they allow that single root node to allocate and manage a RPL Instance, identified by a local RPLInstanceID, without

coordination with other nodes. Typically, this is done in order to optimize routes to a destination within the LLN. (See Section 5).

Global DODAG: A Global DODAG uses a global RPLInstanceID that may be coordinated among several other nodes. (See Section 5).

DIO: DODAG Information Object (see Section 6.3)

DAO: Destination Advertisement Object (see Section 6.4)

DIS: DODAG Information Solicitation (see Section 6.2)

CC: Consistency Check (see Section 6.6)

As they form networks, LLN devices often mix the roles of host and router when compared to traditional IP networks. In this document, "host" refers to an LLN device that can generate but does not forward RPL traffic; "router" refers to an LLN device that can forward as well as generate RPL traffic; and "node" refers to any RPL device, either a host or a router.

3. Protocol Overview

The aim of this section is to describe RPL in the spirit of [RFC4101]. Protocol details can be found in further sections.

3.1. Topologies

This section describes the basic RPL topologies that may be formed, and the rules by which these are constructed, i.e., the rules governing DODAG formation.

3.1.1. Constructing Topologies

LLNs, such as Radio Networks, do not typically have predefined topologies, for example, those imposed by point-to-point wires, so RPL has to discover links and then select peers sparingly.

In many cases, because Layer 2 ranges overlap only partially, RPL forms non-transitive / Non-Broadcast Multi-Access (NBMA) network topologies upon which it computes routes.

RPL routes are optimized for traffic to or from one or more roots that act as sinks for the topology. As a result, RPL organizes a topology as a Directed Acyclic Graph (DAG) that is partitioned into

one or more Destination Oriented DAGs (DODAGs), one DODAG per sink. If the DAG has multiple roots, then it is expected that the roots are federated by a common backbone, such as a transit link.

3.1.2. RPL Identifiers

RPL uses four values to identify and maintain a topology:

- The first is a RPLInstanceID. A RPLInstanceID identifies a set of one or more Destination Oriented DAGs (DODAGs). A network may have multiple RPLInstanceIDs, each of which defines an independent set of DODAGs, which may be optimized for different Objective Functions (OFs) and/or applications. The set of DODAGs identified by a RPLInstanceID is called a RPL Instance. All DODAGs in the same RPL Instance use the same OF.

- The second is a DODAGID. The scope of a DODAGID is a RPL Instance. The combination of RPLInstanceID and DODAGID uniquely identifies a single DODAG in the network. A RPL Instance may have multiple DODAGs, each of which has an unique DODAGID.

- The third is a DODAGVersionNumber. The scope of a DODAGVersionNumber is a DODAG. A DODAG is sometimes reconstructed from the DODAG root, by incrementing the DODAGVersionNumber. The combination of RPLInstanceID, DODAGID, and DODAGVersionNumber uniquely identifies a DODAG Version.

- The fourth is Rank. The scope of Rank is a DODAG Version. Rank establishes a partial order over a DODAG Version, defining individual node positions with respect to the DODAG root.

3.1.3. Instances, DODAGs, and DODAG Versions

A RPL Instance contains one or more DODAG roots. A RPL Instance may provide routes to certain destination prefixes, reachable via the DODAG roots or alternate paths within the DODAG. These roots may operate independently, or they may coordinate over a network that is not necessarily as constrained as an LLN.

A RPL Instance may comprise:

- a single DODAG with a single root
  - For example, a DODAG optimized to minimize latency rooted at a single centralized lighting controller in a Home Automation application.

- multiple uncoordinated DODAGs with independent roots (differing DODAGIDs)
  - For example, multiple data collection points in an urban data collection application that do not have suitable connectivity to coordinate with each other or that use the formation of multiple DODAGs as a means to dynamically and autonomously partition the network.
- a single DODAG with a virtual root that coordinates LLN sinks (with the same DODAGID) over a backbone network.
  - For example, multiple border routers operating with a reliable transit link, e.g., in support of an IPv6 Low-Power Wireless Personal Area Network (6LoWPAN) application, that are capable of acting as logically equivalent interfaces to the sink of the same DODAG.
- a combination of the above as suited to some application scenario.

Each RPL packet is associated with a particular RPLInstanceID (see Section 11.2) and, therefore, RPL Instance (Section 5). The provisioning or automated discovery of a mapping between a RPLInstanceID and a type or service of application traffic is out of scope for this specification (to be defined in future companion specifications).

Figure 1 depicts an example of a RPL Instance comprising three DODAGs with DODAG roots R1, R2, and R3. Each of these DODAG roots advertises the same RPLInstanceID. The lines depict connectivity between parents and children.

Figure 2 depicts how a DODAGVersionNumber increment leads to a new DODAG Version. This depiction illustrates a DODAGVersionNumber increment that results in a different DODAG topology. Note that a new DODAG Version does not always imply a different DODAG topology. To accommodate certain topology changes requires a new DODAG Version, as described later in this specification.

In the following examples, please note that tree-like structures are depicted for simplicity, although the DODAG structure allows for each node to have multiple parents when the connectivity supports it.

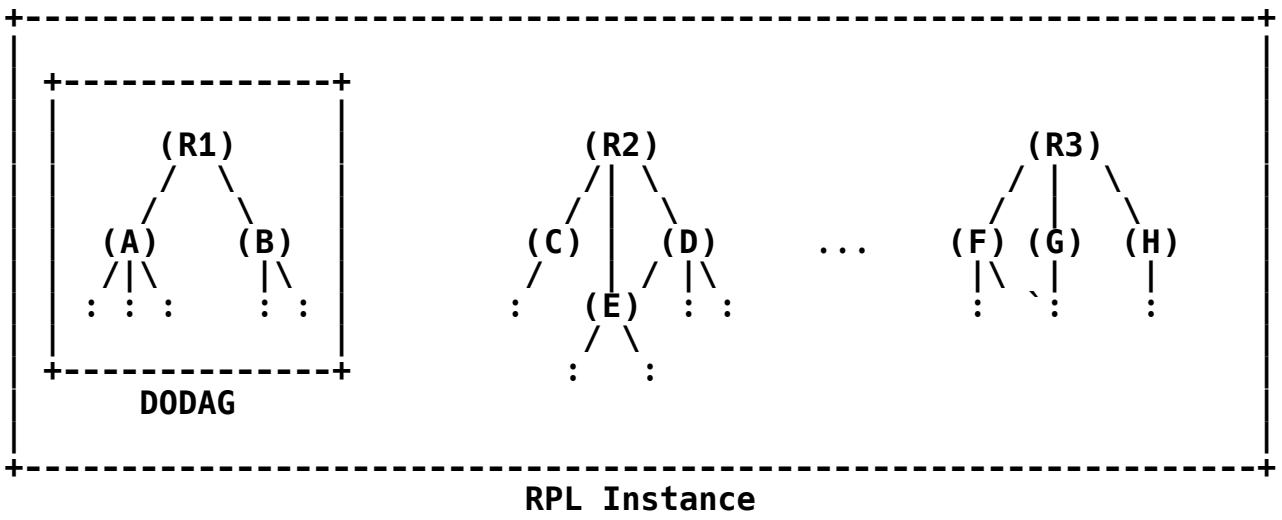


Figure 1: RPL Instance

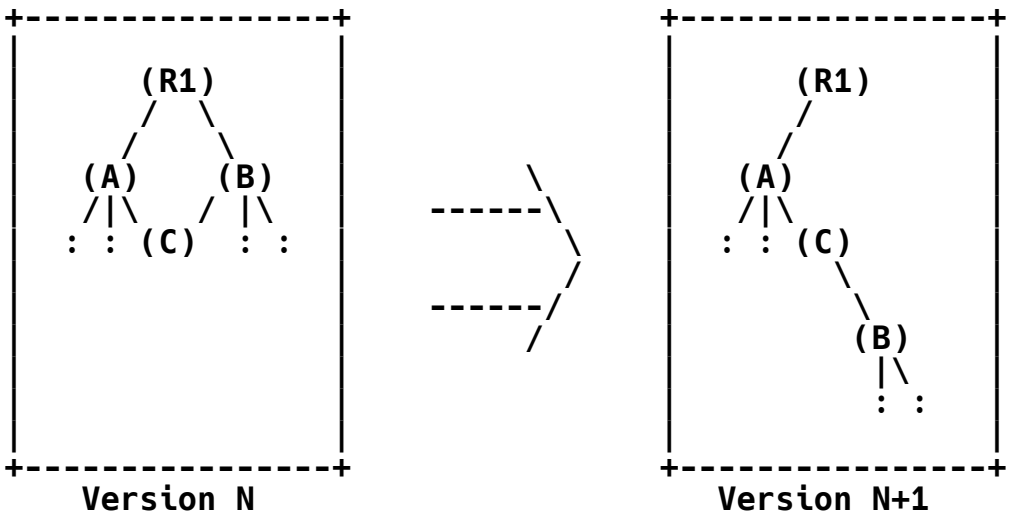


Figure 2: DODAG Version

3.2. Upward Routes and DODAG Construction

RPL provisions routes Up towards DODAG roots, forming a DODAG optimized according to an Objective Function (OF). RPL nodes construct and maintain these DODAGs through DODAG Information Object (DIO) messages.

3.2.1. Objective Function (OF)

The Objective Function (OF) defines how RPL nodes select and optimize routes within a RPL Instance. The OF is identified by an Objective Code Point (OCP) within the DIO Configuration option. An OF defines how nodes translate one or more metrics and constraints, which are themselves defined in [RFC6551], into a value called Rank, which approximates the node's distance from a DODAG root. An OF also defines how nodes select parents. Further details may be found in Section 14, [RFC6551], [RFC6552], and related companion specifications.

3.2.2. DODAG Repair

A DODAG root institutes a global repair operation by incrementing the DODAGVersionNumber. This initiates a new DODAG Version. Nodes in the new DODAG Version can choose a new position whose Rank is not constrained by their Rank within the old DODAG Version.

RPL also supports mechanisms that may be used for local repair within the DODAG Version. The DIO message specifies the necessary parameters as configured from and controlled by policy at the DODAG root.

3.2.3. Security

RPL supports message confidentiality and integrity. It is designed such that link-layer mechanisms can be used when available and appropriate; yet, in their absence, RPL can use its own mechanisms. RPL has three basic security modes.

In the first, called "unsecured", RPL control messages are sent without any additional security mechanisms. Unsecured mode does not imply that the RPL network is unsecure: it could be using other present security primitives (e.g., link-layer security) to meet application security requirements.

In the second, called "preinstalled", nodes joining a RPL Instance have preinstalled keys that enable them to process and generate secured RPL messages.

The third mode is called "authenticated". In authenticated mode, nodes have preinstalled keys as in preinstalled mode, but the preinstalled key may only be used to join a RPL Instance as a leaf. Joining an authenticated RPL Instance as a router requires obtaining a key from an authentication authority. The process by which this key is obtained is out of scope for this specification. Note that this specification alone does not provide sufficient detail for a RPL

implementation to securely operate in authenticated mode. For a RPL implementation to operate securely in authenticated mode, it is necessary for a future companion specification to detail the mechanisms by which a node obtains/requests the authentication material (e.g., key, certificate) and to determine from where that material should be obtained. See also Section 10.3.

3.2.4. Grounded and Floating DODAGs

DODAGs can be grounded or floating: the DODAG root advertises which is the case. A grounded DODAG offers connectivity to hosts that are required for satisfying the application-defined goal. A floating DODAG is not expected to satisfy the goal; in most cases, it only provides routes to nodes within the DODAG. Floating DODAGs may be used, for example, to preserve interconnectivity during repair.

3.2.5. Local DODAGs

RPL nodes can optimize routes to a destination within an LLN by forming a Local DODAG whose DODAG root is the desired destination. Unlike global DAGs, which can consist of multiple DODAGs, local DAGs have one and only one DODAG and therefore one DODAG root. Local DODAGs can be constructed on demand.

3.2.6. Administrative Preference

An implementation/deployment may specify that some DODAG roots should be used over others through an administrative preference. Administrative preference offers a way to control traffic and engineer DODAG formation in order to better support application requirements or needs.

3.2.7. Data-Path Validation and Loop Detection

The low-power and lossy nature of LLNs motivates RPL's use of on-demand loop detection using data packets. Because data traffic can be infrequent, maintaining a routing topology that is constantly up to date with the physical topology can waste energy. Typical LLNs exhibit variations in physical connectivity that are transient and innocuous to traffic, but that would be costly to track closely from the control plane. Transient and infrequent changes in connectivity need not be addressed by RPL until there is data to send. This aspect of RPL's design draws from existing, highly used LLN protocols as well as extensive experimental and deployment evidence on its efficacy.

The RPL Packet Information that is transported with data packets includes the Rank of the transmitter. An inconsistency between the routing decision for a packet (Upward or Downward) and the Rank relationship between the two nodes indicates a possible loop. On receiving such a packet, a node institutes a local repair operation.

For example, if a node receives a packet flagged as moving in the Upward direction, and if that packet records that the transmitter is of a lower (lesser) Rank than the receiving node, then the receiving node is able to conclude that the packet has not progressed in the Upward direction and that the DODAG is inconsistent.

3.2.8. Distributed Algorithm Operation

A high-level overview of the distributed algorithm, which constructs the DODAG, is as follows:

- Some nodes are configured to be DODAG roots, with associated DODAG configurations.
- Nodes advertise their presence, affiliation with a DODAG, routing cost, and related metrics by sending link-local multicast DIO messages to all-RPL-nodes.
- Nodes listen for DIOs and use their information to join a new DODAG (thus, selecting DODAG parents), or to maintain an existing DODAG, according to the specified Objective Function and Rank of their neighbors.
- Nodes provision routing table entries, for the destinations specified by the DIO message, via their DODAG parents in the DODAG Version. Nodes that decide to join a DODAG can provision one or more DODAG parents as the next hop for the default route and a number of other external routes for the associated instance.

3.3. Downward Routes and Destination Advertisement

RPL uses Destination Advertisement Object (DAO) messages to establish Downward routes. DAO messages are an optional feature for applications that require point-to-multipoint (P2MP) or point-to-point (P2P) traffic. RPL supports two modes of Downward traffic: Storing (fully stateful) or Non-Storing (fully source routed); see Section 9. Any given RPL Instance is either storing or non-storing. In both cases, P2P packets travel Up toward a DODAG root then Down to the final destination (unless the destination is on the Upward route). In the Non-Storing case, the packet will travel all the way to a DODAG root before traveling Down. In the Storing case, the

packet may be directed Down towards the destination by a common ancestor of the source and the destination prior to reaching a DODAG root.

As of the writing of this specification, no implementation is expected to support both Storing and Non-Storing modes of operation. Most implementations are expected to support either no Downward routes, Non-Storing mode only, or Storing mode only. Other modes of operation, such as a hybrid mix of Storing and Non-Storing mode, are out of scope for this specification and may be described in other companion specifications.

This specification describes a basic mode of operation in support of P2P traffic. Note that more optimized P2P solutions may be described in companion specifications.

3.4. Local DODAGs Route Discovery

Optionally, a RPL network can support on-demand discovery of DODAGs to specific destinations within an LLN. Such Local DODAGs behave slightly differently than Global DODAGs: they are uniquely defined by the combination of DODAGID and RPLInstanceID. The RPLInstanceID denotes whether a DODAG is a Local DODAG.

3.5. Rank Properties

The Rank of a node is a scalar representation of the location of that node within a DODAG Version. The Rank is used to avoid and detect loops and, as such, must demonstrate certain properties. The exact calculation of the Rank is left to the Objective Function. Even though the specific computation of the Rank is left to the Objective Function, the Rank must implement generic properties regardless of the Objective Function.

In particular, the Rank of the nodes must monotonically decrease as the DODAG Version is followed towards the DODAG destination. In that regard, the Rank can be considered a scalar representation of the location or radius of a node within a DODAG Version.

The details of how the Objective Function computes Rank are out of scope for this specification, although that computation may depend, for example, on parents, link metrics, node metrics, and the node configuration and policies. See Section 14 for more information.

The Rank is not a path cost, although its value can be derived from and influenced by path metrics. The Rank has properties of its own that are not necessarily those of all metrics:

Type: The Rank is an abstract numeric value.

Function: The Rank is the expression of a relative position within a DODAG Version with regard to neighbors, and it is not necessarily a good indication or a proper expression of a distance or a path cost to the root.

Stability: The stability of the Rank determines the stability of the routing topology. Some dampening or filtering is RECOMMENDED to keep the topology stable; thus, the Rank does not necessarily change as fast as some link or node metrics would. A new DODAG Version would be a good opportunity to reconcile the discrepancies that might form over time between metrics and Ranks within a DODAG Version.

Properties: The Rank is incremented in a strictly monotonic fashion, and it can be used to validate a progression from or towards the root. A metric, like bandwidth or jitter, does not necessarily exhibit this property.

Abstract: The Rank does not have a physical unit, but rather a range of increment per hop, where the assignment of each increment is to be determined by the Objective Function.

The Rank value feeds into DODAG parent selection, according to the RPL loop-avoidance strategy. Once a parent has been added, and a Rank value for the node within the DODAG has been advertised, the node's further options with regard to DODAG parent selection and movement within the DODAG are restricted in favor of loop avoidance.

3.5.1. Rank Comparison (DAGRank())

Rank may be thought of as a fixed-point number, where the position of the radix point between the integer part and the fractional part is determined by MinHopRankIncrease. MinHopRankIncrease is the minimum increase in Rank between a node and any of its DODAG parents. A DODAG root provisions MinHopRankIncrease. MinHopRankIncrease creates a trade-off between hop cost precision and the maximum number of hops a network can support. A very large MinHopRankIncrease, for example, allows precise characterization of a given hop's effect on Rank but cannot support many hops.

When an Objective Function computes Rank, the Objective Function operates on the entire (i.e., 16-bit) Rank quantity. When Rank is compared, e.g., for determination of parent relationships or loop detection, the integer portion of the Rank is to be used. The

integer portion of the Rank is computed by the DAGRank() macro as follows, where floor(x) is the function that evaluates to the greatest integer less than or equal to x:

DAGRank(rank) = floor(rank/MinHopRankIncrease)

For example, if a 16-bit Rank quantity is decimal 27, and the MinHopRankIncrease is decimal 16, then DAGRank(27) = floor(1.6875) = 1. The integer part of the Rank is 1 and the fractional part is 11/16.

Following the conventions in this document, using the macro DAGRank(node) may be interpreted as DAGRank(node.rank), where node.rank is the Rank value as maintained by the node.

A Node A has a Rank less than the Rank of a Node B if DAGRank(A) is less than DAGRank(B).

A Node A has a Rank equal to the Rank of a Node B if DAGRank(A) is equal to DAGRank(B).

A Node A has a Rank greater than the Rank of a Node B if DAGRank(A) is greater than DAGRank(B).

3.5.2. Rank Relationships

Rank computations maintain the following properties for any nodes M and N that are neighbors in the LLN:

DAGRank(M) is less than DAGRank(N):

> In this case, the position of M is closer to the DODAG root than the position of N. Node M may safely be a DODAG parent for Node N without risk of creating a loop. Further, for a Node N, all parents in the DODAG parent set must be of a Rank less than DAGRank(N). In other words, the Rank presented by a Node N MUST be greater than that presented by any of its parents.

DAGRank(M) equals DAGRank(N):

> In this case, the positions of M and N within the DODAG and with respect to the DODAG root are similar or identical. Routing through a node with equal Rank may cause a routing loop (i.e., if that node chooses to route through a node with equal Rank as well).

DAGRank(M) is greater than DAGRank(N):

In this case, the position of M is farther from the DODAG root than the position of N. Further, Node M may in fact be in the sub-DODAG of Node N. If Node N selects Node M as DODAG parent, there is a risk of creating a loop.

As an example, the Rank could be computed in such a way so as to closely track ETX (expected transmission count, a fairly common routing metric used in LLN and defined in [RFC6551]) when the metric that an Objective Function minimizes is ETX, or latency, or in a more complicated way as appropriate to the Objective Function being used within the DODAG.

3.6. Routing Metrics and Constraints Used by RPL

Routing metrics are used by routing protocols to compute shortest paths. Interior Gateway Protocols (IGPs) such as IS-IS ([RFC5120]) and OSPF ([RFC4915]) use static link metrics. Such link metrics can simply reflect the bandwidth or can also be computed according to a polynomial function of several metrics defining different link characteristics. Some routing protocols support more than one metric: in the vast majority of the cases, one metric is used per (sub-)topology. Less often, a second metric may be used as a tiebreaker in the presence of Equal Cost Multiple Paths (ECMPs). The optimization of multiple metrics is known as an NP-complete problem and is sometimes supported by some centralized path computation engine.

In contrast, LLNs do require the support of both static and dynamic metrics. Furthermore, both link and node metrics are required. In the case of RPL, it is virtually impossible to define one metric, or even a composite metric, that will satisfy all use cases.

In addition, RPL supports constraint-based routing where constraints may be applied to both link and nodes. If a link or a node does not satisfy a required constraint, it is "pruned" from the candidate neighbor set, thus leading to a constrained shortest path.

An Objective Function specifies the objectives used to compute the (constrained) path. Furthermore, nodes are configured to support a set of metrics and constraints and select their parents in the DODAG according to the metrics and constraints advertised in the DIO messages. Upstream and Downstream metrics may be merged or advertised separately depending on the OF and the metrics. When they are advertised separately, it may happen that the set of DIO parents

is different from the set of DAO parents (a DAO parent is a node to which unicast DAO messages are sent). Yet, all are DODAG parents with regard to the rules for Rank computation.

The Objective Function is decoupled from the routing metrics and constraints used by RPL. Whereas the OF dictates rules such as DODAG parent selection, load balancing, and so on, the set of metrics and/or constraints used, and thus those that determine the preferred path, are based on the information carried within the DAG container option in DIO messages.

The set of supported link/node constraints and metrics is specified in [RFC6551].

Example 1: Shortest path: path offering the shortest end-to-end delay.

Example 2: Shortest Constrained path: the path that does not traverse any battery-operated node and that optimizes the path reliability.

3.7. Loop Avoidance

RPL tries to avoid creating loops when undergoing topology changes and includes Rank-based data-path validation mechanisms for detecting loops when they do occur (see Section 11 for more details). In practice, this means that RPL guarantees neither loop-free path selection nor tight delay convergence times, but it can detect and repair a loop as soon as it is used. RPL uses this loop detection to ensure that packets make forward progress within the DODAG Version and trigger repairs when necessary.

3.7.1. Greediness and Instability

A node is greedy if it attempts to move deeper (increase Rank) in the DODAG Version in order to increase the size of the parent set or improve some other metric. Once a node has joined a DODAG Version, RPL disallows certain behaviors, including greediness, in order to prevent resulting instabilities in the DODAG Version.

Suppose a node is willing to receive and process a DIO message from a node in its own sub-DODAG and, in general, a node deeper than itself. In this case, a possibility exists that a feedback loop is created, wherein two or more nodes continue to try and move in the DODAG Version while attempting to optimize against each other. In some cases, this will result in instability. It is for this reason that RPL limits the cases where a node may process DIO messages from deeper nodes to some form of local repair. This approach creates an

"event horizon", whereby a node cannot be influenced beyond some limit into an instability by the action of nodes that may be in its own sub-DODAG.

3.7.1.1. Example: Greedy Parent Selection and Instability

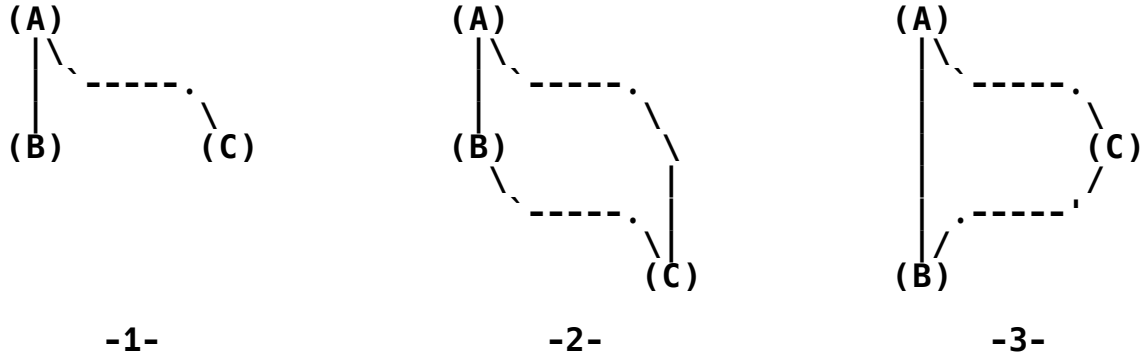


Figure 3: Greedy DODAG Parent Selection

Figure 3 depicts a DODAG in three different configurations. A usable link between (B) and (C) exists in all three configurations. In Figure 3-1, Node (A) is a DODAG parent for Nodes (B) and (C). In Figure 3-2, Node (A) is a DODAG parent for Nodes (B) and (C), and Node (B) is also a DODAG parent for Node (C). In Figure 3-3, Node (A) is a DODAG parent for Nodes (B) and (C), and Node (C) is also a DODAG parent for Node (B).

If a RPL node is too greedy, in that it attempts to optimize for an additional number of parents beyond its most preferred parents, then an instability can result. Consider the DODAG illustrated in Figure 3-1. In this example, Nodes (B) and (C) may most prefer Node (A) as a DODAG parent, but we will consider the case when they are operating under the greedy condition that will try to optimize for two parents.

- Let Figure 3-1 be the initial condition.
- Suppose Node (C) first is able to leave the DODAG and rejoin at a lower Rank, taking both Nodes (A) and (B) as DODAG parents as depicted in Figure 3-2. Now Node (C) is deeper than both Nodes (A) and (B), and Node (C) is satisfied to have two DODAG parents.
- Suppose Node (B), in its greediness, is willing to receive and process a DIO message from Node (C) (against the rules of RPL), and then Node (B) leaves the DODAG and rejoins at a lower Rank,

taking both Nodes (A) and (C) as DODAG parents. Now Node (B) is deeper than both Nodes (A) and (C) and is satisfied with two DAG parents.

- Then, Node (C), because it is also greedy, will leave and rejoin deeper, to again get two parents and have a lower Rank then both of them.
- Next, Node (B) will again leave and rejoin deeper, to again get two parents.
- Again, Node (C) leaves and rejoins deeper.
- The process will repeat, and the DODAG will oscillate between Figure 3-2 and Figure 3-3 until the nodes count to infinity and restart the cycle again.
- This cycle can be averted through mechanisms in RPL:
  - Nodes (B) and (C) stay at a Rank sufficient to attach to their most preferred parent (A) and don't go for any deeper (worse) alternate parents (Nodes are not greedy).
  - Nodes (B) and (C) do not process DIO messages from nodes deeper than themselves (because such nodes are possibly in their own sub-DODAGs).

These mechanisms are further described in Section 8.2.2.4.

3.7.2. DODAG Loops

A DODAG loop may occur when a node detaches from the DODAG and reattaches to a device in its prior sub-DODAG. In particular, this may happen when DIO messages are missed. Strict use of the DODAGVersionNumber can eliminate this type of loop, but this type of loop may possibly be encountered when using some local repair mechanisms.

For example, consider the local repair mechanism that allows a node to detach from the DODAG, advertise a Rank of INFINITE_RANK (in order to poison its routes / inform its sub-DODAG), and then reattach to the DODAG. In some of these cases, the node may reattach to its own prior-sub-DODAG, causing a DODAG loop, because the poisoning may fail if the INFINITE_RANK advertisements are lost in the LLN environment. (In this case, the Rank-based data-path validation mechanisms would eventually detect and trigger correction of the loop).

3.7.3. DAO Loops

A DAO loop may occur when the parent has a route installed upon receiving and processing a DAO message from a child, but the child has subsequently cleaned up the related DAO state. This loop happens when a No-Path (a DAO message that invalidates a previously announced prefix, see Section 6.4.3) was missed and persists until all state has been cleaned up. RPL includes an optional mechanism to acknowledge DAO messages, which may mitigate the impact of a single DAO message being missed. RPL includes loop detection mechanisms that mitigate the impact of DAO loops and trigger their repair. (See Section 11.2.2.3.)

4. Traffic Flows Supported by RPL

RPL supports three basic traffic flows: multipoint-to-point (MP2P), point-to-multipoint (P2MP), and point-to-point (P2P).

4.1. Multipoint-to-Point Traffic

Multipoint-to-point (MP2P) is a dominant traffic flow in many LLN applications ([RFC5867], [RFC5826], [RFC5673], and [RFC5548]). The destinations of MP2P flows are designated nodes that have some application significance, such as providing connectivity to the larger Internet or core private IP network. RPL supports MP2P traffic by allowing MP2P destinations to be reached via DODAG roots.

4.2. Point-to-Multipoint Traffic

Point-to-multipoint (P2MP) is a traffic pattern required by several LLN applications ([RFC5867], [RFC5826], [RFC5673], and [RFC5548]). RPL supports P2MP traffic by using a destination advertisement mechanism that provisions Down routes toward destinations (prefixes, addresses, or multicast groups), and away from roots. Destination advertisements can update routing tables as the underlying DODAG topology changes.

4.3. Point-to-Point Traffic

RPL DODAGs provide a basic structure for point-to-point (P2P) traffic. For a RPL network to support P2P traffic, a root must be able to route packets to a destination. Nodes within the network may also have routing tables to destinations. A packet flows towards a root until it reaches an ancestor that has a known route to the destination. As pointed out later in this document, in the most constrained case (when nodes cannot store routes), that common ancestor may be the DODAG root. In other cases, it may be a node closer to both the source and destination.

RPL also supports the case where a P2P destination is a 'one-hop' neighbor.

RPL neither specifies nor precludes additional mechanisms for computing and installing potentially more optimal routes to support arbitrary P2P traffic.

5. RPL Instance

Within a given LLN, there may be multiple, logically independent RPL Instances. A RPL node may belong to multiple RPL Instances, and it may act as a router in some and as a leaf in others. This document describes how a single instance behaves.

There are two types of RPL Instances: Local and Global. RPL divides the RPLInstanceID space between Global and Local instances to allow for both coordinated and unilateral allocation of RPLInstanceIDs. Global RPL Instances are coordinated, have one or more DODAGs, and are typically long-lived. Local RPL Instances are always a single DODAG whose singular root owns the corresponding DODAGID and allocates the local RPLInstanceID in a unilateral manner. Local RPL Instances can be used, for example, for constructing DODAGs in support of a future on-demand routing solution. The mode of operation of Local RPL Instances is out of scope for this specification and may be described in other companion specifications.

The definition and provisioning of RPL Instances are out of scope for this specification. Guidelines may be application and implementation specific, and they are expected to be elaborated in future companion specifications. Those operations are expected to be such that data packets coming from the outside of the RPL network can unambiguously be associated to at least one RPL Instance and be safely routed over any instance that would match the packet.

Control and data packets within RPL network are tagged to unambiguously identify of which RPL Instance they are a part.

Every RPL control message has a RPLInstanceID field. Some RPL control messages, when referring to a local RPLInstanceID as defined below, may also include a DODAGID.

Data packets that flow within the RPL network expose the RPLInstanceID as part of the RPL Packet Information that RPL requires, as further described in Section 11.2. For data packets coming from outside the RPL network, the ingress router determines the RPLInstanceID and places it into the resulting packet that it injects into the RPL network.

5.1. RPL Instance ID

A global RPLInstanceID MUST be unique to the whole LLN. Mechanisms for allocating and provisioning global RPLInstanceID are out of scope for this specification. There can be up to 128 Global instance in the whole network. Local instances are always used in conjunction with a DODAGID (which is either given explicitly or implicitly in some cases), and up 64 Local instances per DODAGID can be supported. Local instances are allocated and managed by the node that owns the DODAGID, without any explicit coordination with other nodes, as further detailed below.

A global RPLInstanceID is encoded in a RPLInstanceID field as follows:

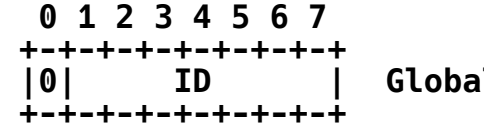


Figure 4: RPLInstanceID Field Format for Global Instances

A local RPLInstanceID is autoconfigured by the node that owns the DODAGID and it MUST be unique for that DODAGID. The DODAGID used to configure the local RPLInstanceID MUST be a reachable IPv6 address of the node, and it MUST be used as an endpoint of all communications within that Local instance.

A local RPLInstanceID is encoded in a RPLInstanceID field as follows:

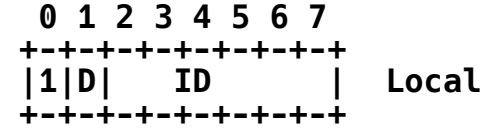


Figure 5: RPLInstanceID Field Format for Local Instances

The 'D' flag in a local RPLInstanceID is always set to 0 in RPL control messages. It is used in data packets to indicate whether the DODAGID is the source or the destination of the packet. If the 'D' flag is set to 1, then the destination address of the IPv6 packet MUST be the DODAGID. If the 'D' flag is cleared, then the source address of the IPv6 packet MUST be the DODAGID.

For example, consider a Node A that is the DODAG root of a Local RPL Instance, and has allocated a local RPLInstanceID. By definition, all traffic traversing that Local RPL Instance will either originate or terminate at Node A. In this case, the DODAGID will be the

reachable IPv6 address of Node A. All traffic will contain the address of Node A, and thus the DODAGID, in either the source or destination address. Thus, the local RPLInstanceID may indicate that the DODAGID is equivalent to either the source address or the destination address by setting the 'D' flag appropriately.

6. ICMPv6 RPL Control Message

This document defines the RPL control message, a new ICMPv6 [RFC4443] message. A RPL control message is identified by a code and composed of a base that depends on the code (and a series of options).

Most RPL control messages have the scope of a link. The only exception is for the DAO / DAO-ACK messages in Non-Storing mode, which are exchanged using a unicast address over multiple hops and thus uses global or unique-local addresses for both the source and destination addresses. For all other RPL control messages, the source address is a link-local address, and the destination address is either the all-RPL-nodes multicast address or a link-local unicast address of the destination. The all-RPL-nodes multicast address is a new address with a value of ff02::1a.

In accordance with [RFC4443], the RPL Control Message consists of an ICMPv6 header followed by a message body. The message body is comprised of a message base and possibly a number of options as illustrated in Figure 6.

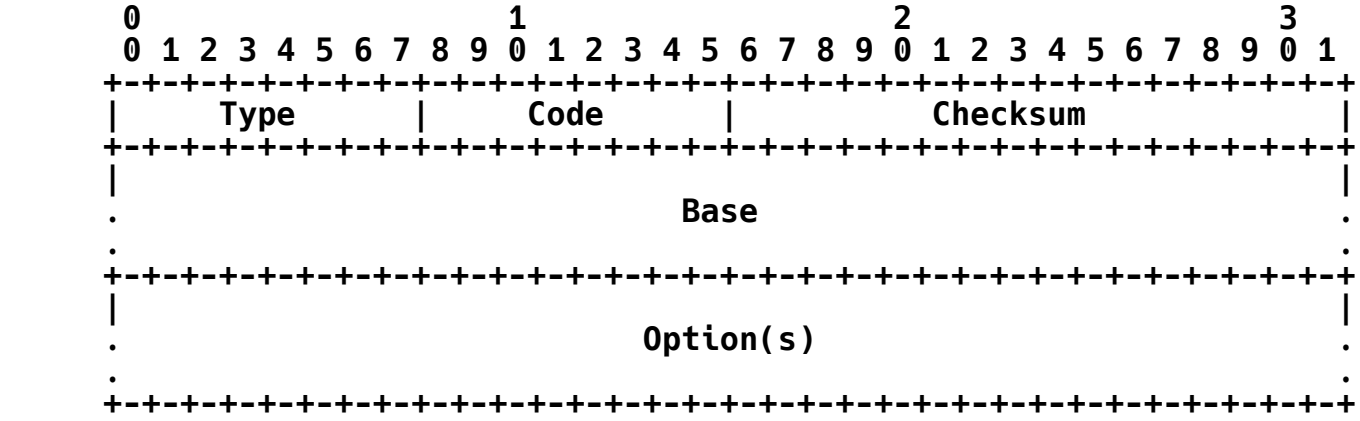


Figure 6: RPL Control Message

The RPL control message is an ICMPv6 information message with a Type of 155.

The Code field identifies the type of RPL control message. This document defines codes for the following RPL control message types (see Section 20.2)):

- 0x00: DODAG Information Solicitation (Section 6.2)
- 0x01: DODAG Information Object (Section 6.3)
- 0x02: Destination Advertisement Object (Section 6.4)
- 0x03: Destination Advertisement Object Acknowledgment (Section 6.5)
- 0x80: Secure DODAG Information Solicitation (Section 6.2.2)
- 0x81: Secure DODAG Information Object (Section 6.3.2)
- 0x82: Secure Destination Advertisement Object (Section 6.4.2)
- 0x83: Secure Destination Advertisement Object Acknowledgment (Section 6.5.2)
- 0x8A: Consistency Check (Section 6.6)

If a node receives a RPL control message with an unknown Code field, the node MUST discard the message without any further processing, MAY raise a management alert, and MUST NOT send any messages in response.

The checksum is computed as specified in [RFC4443]. It is set to zero for the RPL security operations specified below and computed once the rest of the content of the RPL message including the security fields is all set.

The high order bit (0x80) of the code denotes whether the RPL message has security enabled. Secure RPL messages have a format to support confidentiality and integrity, illustrated in Figure 7.

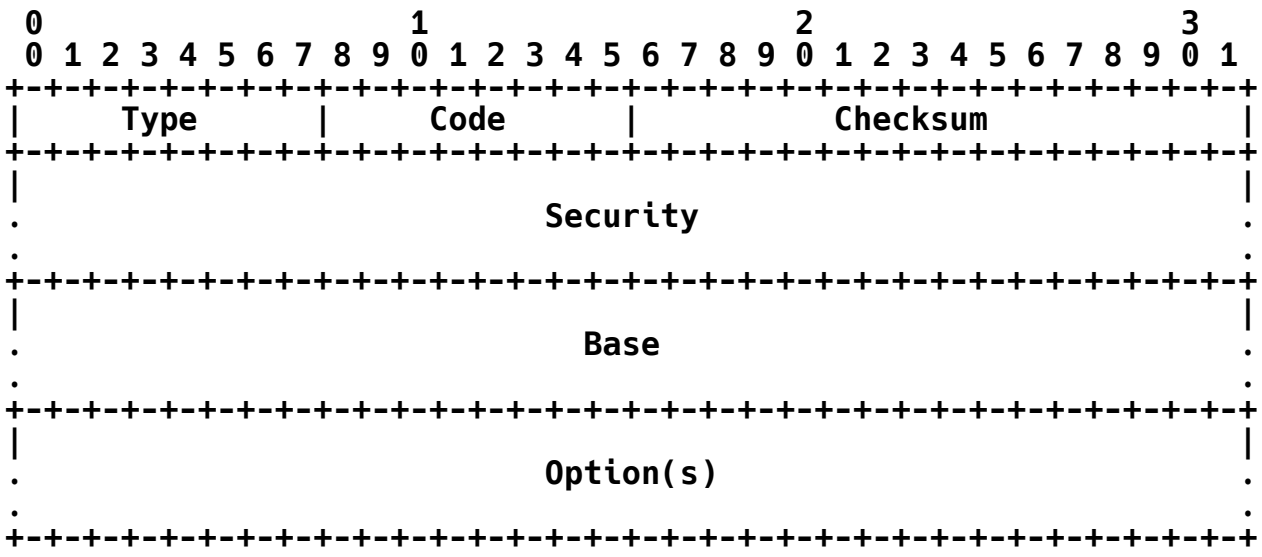


Figure 7: Secure RPL Control Message

The remainder of this section describes the currently defined RPL control message Base formats followed by the currently defined RPL Control Message options.

6.1. RPL Security Fields

Each RPL message has a secure variant. The secure variants provide integrity and replay protection as well as optional confidentiality and delay protection. Because security covers the base message as well as options, in secured messages the security information lies between the checksum and base, as shown in Figure 7.

The level of security and the algorithms in use are indicated in the protocol messages as described below:

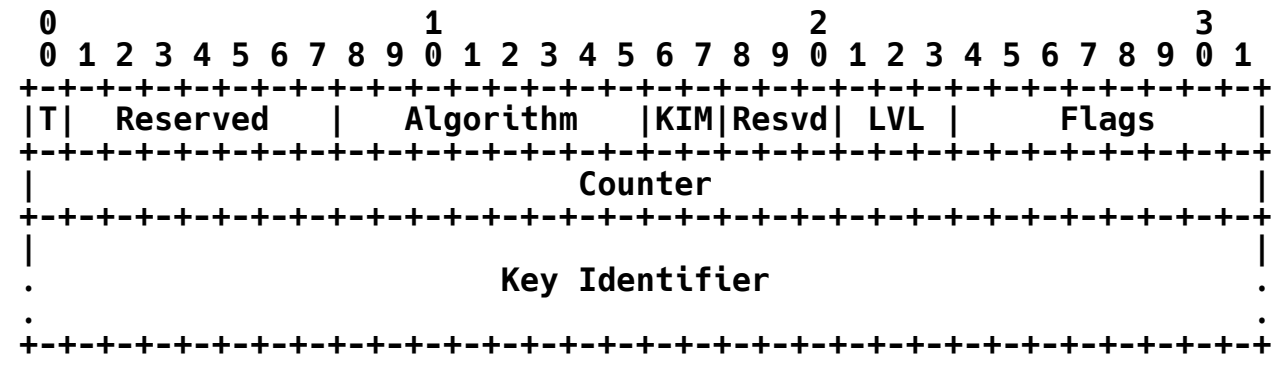


Figure 8: Security Section

Message Authentication Codes (MACs) and signatures provide authentication over the entire unsecured ICMPv6 RPL control message, including the Security section with all fields defined, but with the ICMPv6 checksum temporarily set to zero. Encryption provides confidentiality of the secured RPL ICMPv6 message starting at the first byte after the Security section and continuing to the last byte of the packet. The security transformation yields a secured ICMPv6 RPL message with the inclusion of the cryptographic fields (MAC, signature, etc.). In other words, the security transformation itself (e.g., the Signature and/or Algorithm in use) will detail how to incorporate the cryptographic fields into the secured packet. The Security section itself does not explicitly carry those cryptographic fields. Use of the Security section is further detailed in Sections 19 and 10.

Counter is Time (T): If the counter's Time flag is set, then the Counter field is a timestamp. If the flag is cleared, then the counter is an incrementing counter. Section 10.5 describes the details of the 'T' flag and Counter field.

Reserved: 7-bit unused field. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Security Algorithm (Algorithm): The Security Algorithm field specifies the encryption, MAC, and signature scheme the network uses. Supported values of this field are as follows:

Algorithm	Encryption/MAC	Signature
0	CCM with AES-128	RSA with SHA-256
1-255	Unassigned	Unassigned

Figure 9: Security Algorithm (Algorithm) Encoding

Section 10.9 describes the algorithms in greater detail.

Key Identifier Mode (KIM): The Key Identifier Mode is a 2-bit field that indicates whether the key used for packet protection is determined implicitly or explicitly and indicates the particular representation of the Key Identifier field. The Key Identifier Mode is set one of the values from the table below:

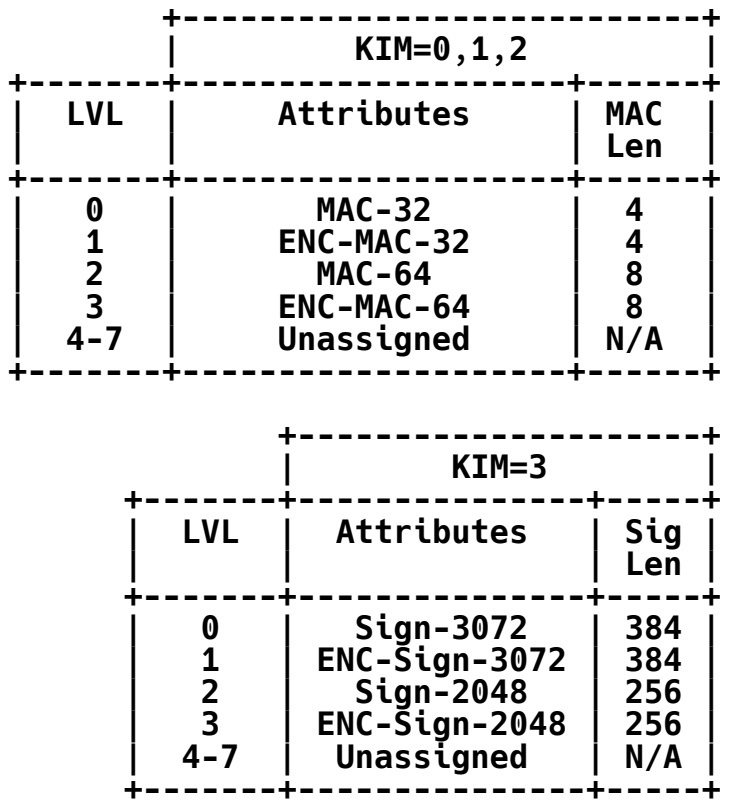
Mode	KIM	Meaning	Key Identifier Length (octets)
0	00	Group key used. Key determined by Key Index field. Key Source is not present. Key Index is present.	1
1	01	Per-pair key used. Key determined by source and destination of packet. Key Source is not present. Key Index is not present.	0
2	10	Group key used. Key determined by Key Index and Key Source Identifier. Key Source is present. Key Index is present.	9
3	11	Node's signature key used. If packet is encrypted, it uses a group key, Key Index and Key Source specify key. Key Source may be present. Key Index may be present.	0/9

Figure 10: Key Identifier Mode (KIM) Encoding

In Mode 3 (KIM=11), the presence or absence of the Key Source and Key Identifier depends on the Security Level (LVL) described below. If the Security Level indicates there is encryption, then the fields are present; if it indicates there is no encryption, then the fields are not present.

Resvd: 3-bit unused field. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Security Level (LVL): The Security Level is a 3-bit field that indicates the provided packet protection. This value can be adapted on a per-packet basis and allows for varying levels of data authenticity and, optionally, for data confidentiality. The KIM field indicates whether signatures are used and the meaning of the Level field. Note that the assigned values of Security Level are not necessarily ordered -- a higher value of LVL does not necessarily equate to increased security. The Security Level is set to one of the values in the tables below:



KIM=0,1,2		
LVL	Attributes	MAC Len
0	MAC-32	4
1	ENC-MAC-32	4
2	MAC-64	8
3	ENC-MAC-64	8
4-7	Unassigned	N/A

KIM=3		
LVL	Attributes	Sig Len
0	Sign-3072	384
1	ENC-Sign-3072	384
2	Sign-2048	256
3	ENC-Sign-2048	256
4-7	Unassigned	N/A

Figure 11: Security Level (LVL) Encoding

The MAC attribute indicates that the message has a MAC of the specified length. The ENC attribute indicates that the message is encrypted. The Sign attribute indicates that the message has a signature of the specified length.

Flags: 8-bit unused field reserved for flags. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Counter: The Counter field indicates the non-repeating 4-octet value used to construct the cryptographic mechanism that implements packet protection and allows for the provision of semantic security. See Section 10.9.1.

Key Identifier: The Key Identifier field indicates which key was used to protect the packet. This field provides various levels of granularity of packet protection, including peer-to-peer keys, group keys, and signature keys. This field is represented as indicated by the Key Identifier Mode field and is formatted as follows:

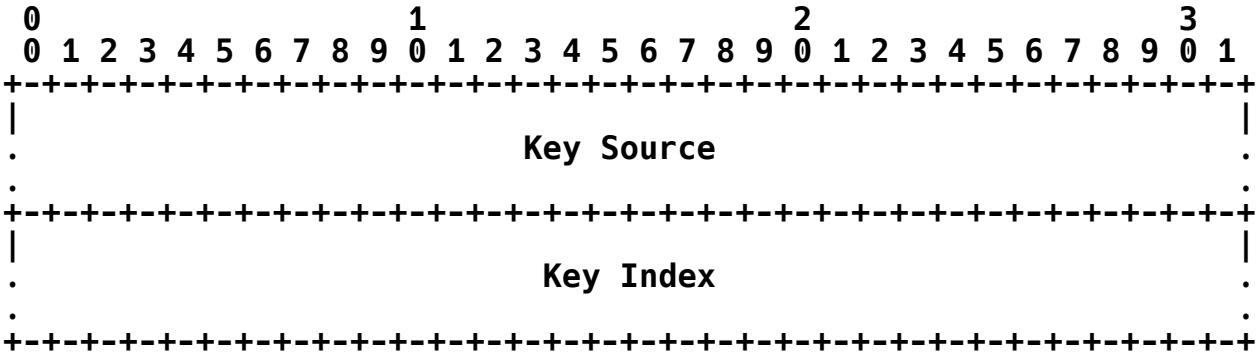


Figure 12: Key Identifier

Key Source: The Key Source field, when present, indicates the logical identifier of the originator of a group key. When present, this field is 8 bytes in length.

Key Index: The Key Index field, when present, allows unique identification of different keys with the same originator. It is the responsibility of each key originator to make sure that actively used keys that it issues have distinct key indices and that all key indices have a value unequal to 0x00. Value 0x00 is reserved for a preinstalled, shared key. When present this field is 1 byte in length.

Unassigned bits of the Security section are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

6.2. DODAG Information Solicitation (DIS)

The DODAG Information Solicitation (DIS) message may be used to solicit a DODAG Information Object from a RPL node. Its use is analogous to that of a Router Solicitation as specified in IPv6 Neighbor Discovery; a node may use DIS to probe its neighborhood for nearby DODAGs. Section 8.3 describes how nodes respond to a DIS.

6.2.1. Format of the DIS Base Object

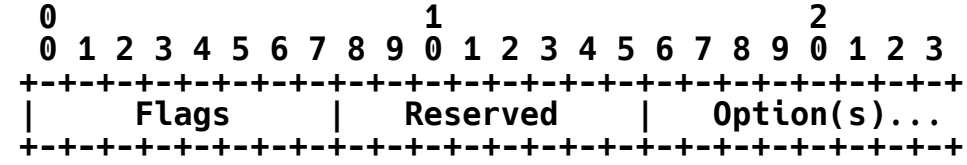


Figure 13: The DIS Base Object

Flags: 8-bit unused field reserved for flags. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Reserved: 8-bit unused field. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Unassigned bits of the DIS Base are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

6.2.2. Secure DIS

A Secure DIS message follows the format in Figure 7, where the base format is the DIS message shown in Figure 13.

6.2.3. DIS Options

The DIS message MAY carry valid options.

This specification allows for the DIS message to carry the following options:

- 0x00 Pad1
- 0x01 PadN
- 0x07 Solicited Information

6.3. DODAG Information Object (DIO)

The DODAG Information Object carries information that allows a node to discover a RPL Instance, learn its configuration parameters,

select a DODAG parent set, and maintain the DODAG.

6.3.1. Format of the DIO Base Object

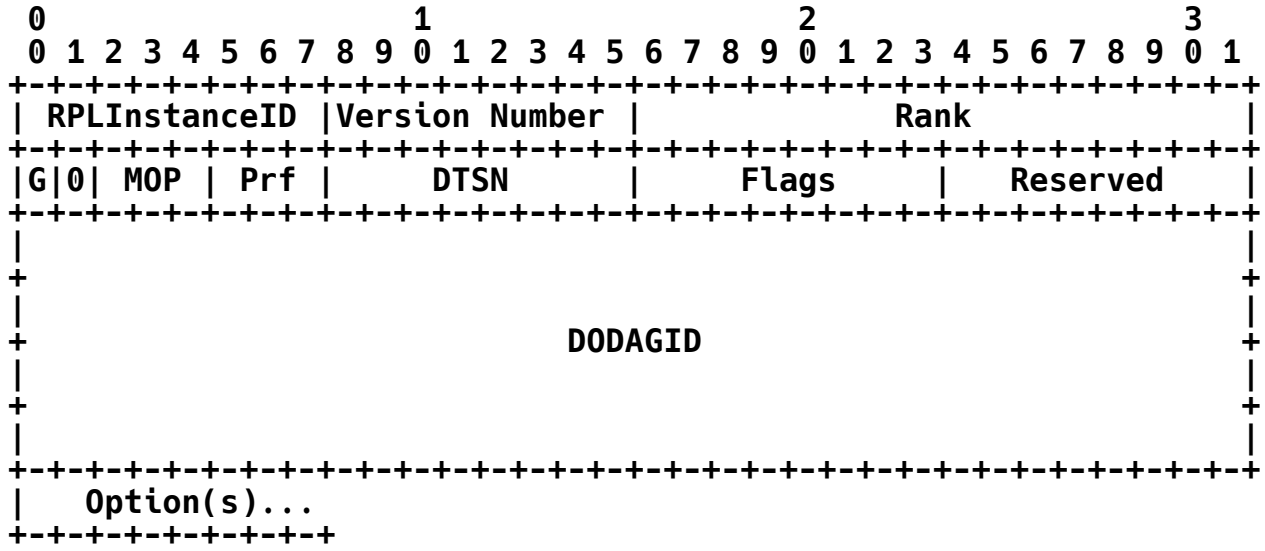


Figure 14: The DIO Base Object

Grounded (G): The Grounded 'G' flag indicates whether the DODAG advertised can satisfy the application-defined goal. If the flag is set, the DODAG is grounded. If the flag is cleared, the DODAG is floating.

Mode of Operation (MOP): The Mode of Operation (MOP) field identifies the mode of operation of the RPL Instance as administratively provisioned at and distributed by the DODAG root. All nodes who join the DODAG must be able to honor the MOP in order to fully participate as a router, or else they must only join as a leaf. MOP is encoded as in the figure below:

MOP	Description
0	No Downward routes maintained by RPL
1	Non-Storing Mode of Operation
2	Storing Mode of Operation with no multicast support
3	Storing Mode of Operation with multicast support
	All other values are unassigned

A value of 0 indicates that destination advertisement messages are disabled and the DODAG maintains only Upward routes.

Figure 15: Mode of Operation (MOP) Encoding

DODAGPreference (Prf): A 3-bit unsigned integer that defines how preferable the root of this DODAG is compared to other DODAG roots within the instance. DAGPreference ranges from 0x00 (least preferred) to 0x07 (most preferred). The default is 0 (least preferred). Section 8.2 describes how DAGPreference affects DIO processing.

Version Number: 8-bit unsigned integer set by the DODAG root to the DODAGVersionNumber. Section 8.2 describes the rules for DODAGVersionNumbers and how they affect DIO processing.

Rank: 16-bit unsigned integer indicating the DODAG Rank of the node sending the DIO message. Section 8.2 describes how Rank is set and how it affects DIO processing.

RPLInstanceID: 8-bit field set by the DODAG root that indicates of which RPL Instance the DODAG is a part.

Destination Advertisement Trigger Sequence Number (DTSN): 8-bit unsigned integer set by the node issuing the DIO message. The Destination Advertisement Trigger Sequence Number (DTSN) flag is used as part of the procedure to maintain Downward routes. The details of this process are described in Section 9.

Flags: 8-bit unused field reserved for flags. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Reserved: 8-bit unused field. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

DODAGID: 128-bit IPv6 address set by a DODAG root that uniquely identifies a DODAG. The DODAGID MUST be a routable IPv6 address belonging to the DODAG root.

Unassigned bits of the DIO Base are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

6.3.2. Secure DIO

A Secure DIO message follows the format in Figure 7, where the base format is the DIO message shown in Figure 14.

6.3.3. DIO Options

The DIO message MAY carry valid options.

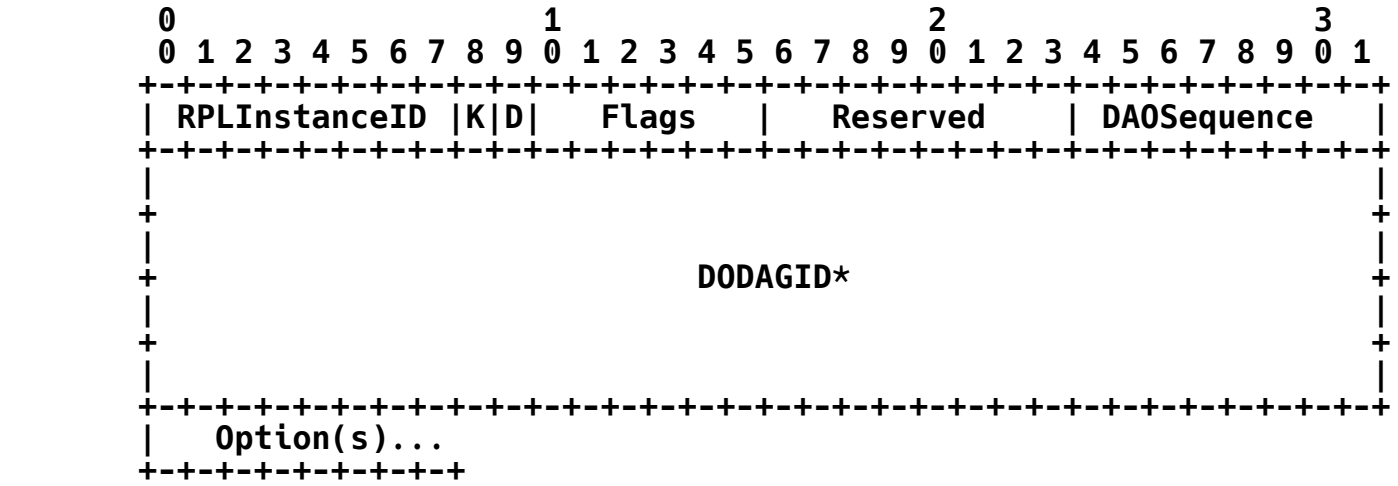
This specification allows for the DIO message to carry the following options:

0x00 Pad1
0x01 PadN
0x02 DAG Metric Container
0x03 Routing Information
0x04 DODAG Configuration
0x08 Prefix Information

6.4. Destination Advertisement Object (DAO)

The Destination Advertisement Object (DAO) is used to propagate destination information Upward along the DODAG. In Storing mode, the DAO message is unicast by the child to the selected parent(s). In Non-Storing mode, the DAO message is unicast to the DODAG root. The DAO message may optionally, upon explicit request or error, be acknowledged by its destination with a Destination Advertisement Acknowledgement (DAO-ACK) message back to the sender of the DAO.

6.4.1. Format of the DAO Base Object



The '*' denotes that the DODAGID is not always present, as described below.

Figure 16: The DAO Base Object

RPLInstanceID: 8-bit field indicating the topology instance associated with the DODAG, as learned from the DIO.

K: The 'K' flag indicates that the recipient is expected to send a DAO-ACK back. (See Section 9.3.)

D: The 'D' flag indicates that the DODAGID field is present. This flag MUST be set when a local RPLInstanceID is used.

Flags: The 6 bits remaining unused in the Flags field are reserved for flags. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Reserved: 8-bit unused field. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

DAOSequence: Incremented at each unique DAO message from a node and echoed in the DAO-ACK message.

DODAGID (optional): 128-bit unsigned integer set by a DODAG root that uniquely identifies a DODAG. This field is only present when the 'D' flag is set. This field is typically only present when a local RPLInstanceID is in use, in order to identify the DODAGID that is associated with the RPLInstanceID. When a global RPLInstanceID is in use, this field need not be present.

Unassigned bits of the DAO Base are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

6.4.2. Secure DAO

A Secure DAO message follows the format in Figure 7, where the base format is the DAO message shown in Figure 16.

6.4.3. DAO Options

The DAO message MAY carry valid options.

This specification allows for the DAO message to carry the following options:

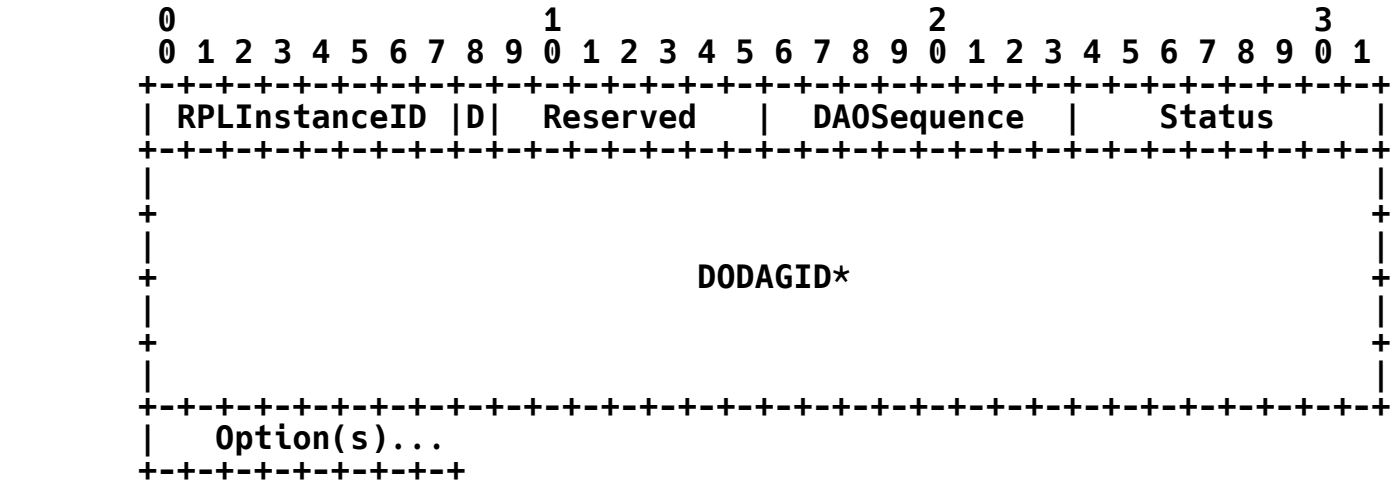
- 0x00 Pad1
- 0x01 PadN
- 0x05 RPL Target
- 0x06 Transit Information
- 0x09 RPL Target Descriptor

A special case of the DAO message, termed a No-Path, is used in Storing mode to clear Downward routing state that has been provisioned through DAO operation. The No-Path carries a Target option and an associated Transit Information option with a lifetime of 0x00000000 to indicate a loss of reachability to that Target.

6.5. Destination Advertisement Object Acknowledgement (DAO-ACK)

The DAO-ACK message is sent as a unicast packet by a DAO recipient (a DAO parent or DODAG root) in response to a unicast DAO message.

6.5.1. Format of the DAO-ACK Base Object



The '*' denotes that the DODAGID is not always present, as described below.

Figure 17: The DAO ACK Base Object

RPLInstanceID: 8-bit field indicating the topology instance associated with the DODAG, as learned from the DIO.

D: The 'D' flag indicates that the DODAGID field is present. This would typically only be set when a local RPLInstanceID is used.

Reserved: The 7-bit field, reserved for flags.

DAOSequence: Incremented at each DAO message from a node, and echoed in the DAO-ACK by the recipient. The DAOSequence is used to correlate a DAO message and a DAO ACK message and is not to be confused with the Transit Information option Path Sequence that is associated to a given Target Down the DODAG.

Status: Indicates the completion. Status 0 is defined as unqualified acceptance in this specification. The remaining status values are reserved as rejection codes. No rejection status codes are defined in this specification, although status codes SHOULD be allocated according to the following guidelines in future specifications:

- 0: Unqualified acceptance (i.e., the node receiving the DAO-ACK is not rejected).

1-127: Not an outright rejection; the node sending the DAO-ACK is willing to act as a parent, but the receiving node is suggested to find and use an alternate parent instead.

127-255: Rejection; the node sending the DAO-ACK is unwilling to act as a parent.

DODAGID (optional): 128-bit unsigned integer set by a DODAG root that uniquely identifies a DODAG. This field is only present when the 'D' flag is set. Typically, this field is only present when a local RPLInstanceID is in use in order to identify the DODAGID that is associated with the RPLInstanceID. When a global RPLInstanceID is in use, this field need not be present.

Unassigned bits of the DAO-ACK Base are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

6.5.2. Secure DAO-ACK

A Secure DAO-ACK message follows the format in Figure 7, where the base format is the DAO-ACK message shown in Figure 17.

6.5.3. DAO-ACK Options

This specification does not define any options to be carried by the DAO-ACK message.

6.6. Consistency Check (CC)

The CC message is used to check secure message counters and issue challenge-responses. A CC message MUST be sent as a secured RPL message.

6.6.1. Format of the CC Base Object

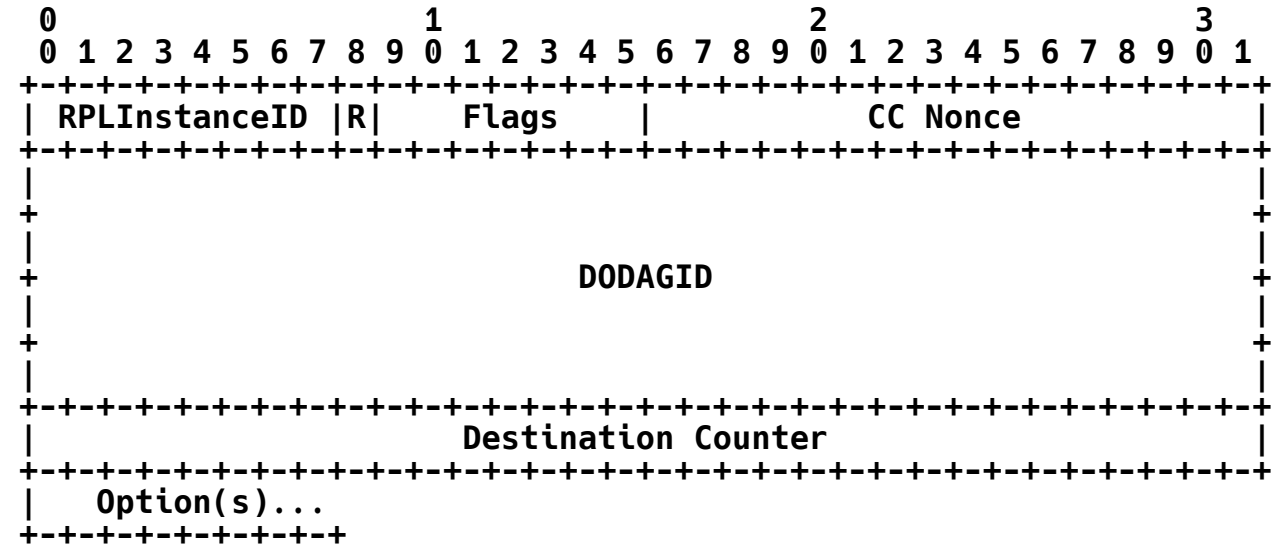


Figure 18: The CC Base Object

RPLInstanceID: 8-bit field indicating the topology instance associated with the DODAG, as learned from the DIO.

R: The 'R' flag indicates whether the CC message is a response. A message with the 'R' flag cleared is a request; a message with the 'R' flag set is a response.

Flags: The 7 bits remaining unused in the Flags field are reserved for flags. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

CC Nonce: 16-bit unsigned integer set by a CC request. The corresponding CC response includes the same CC nonce value as the request.

DODAGID: 128-bit field, contains the identifier of the DODAG root.

Destination Counter: 32-bit unsigned integer value indicating the sender's estimate of the destination's current security counter value. If the sender does not have an estimate, it SHOULD set the Destination Counter field to zero.

Unassigned bits of the CC Base are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

The Destination Counter value allows new or recovered nodes to resynchronize through CC message exchanges. This is important to ensure that a Counter value is not repeated for a given security key even in the event of devices recovering from a failure that created a loss of Counter state. For example, where a CC request or other RPL message is received with an initialized counter within the message Security section, the provision of the Incoming Counter within the CC response message allows the requesting node to reset its Outgoing Counter to a value greater than the last value received by the responding node; the Incoming Counter will also be updated from the received CC response.

6.6.2. CC Options

This specification allows for the CC message to carry the following options:

- 0x00 Pad1
- 0x01 PadN

6.7. RPL Control Message Options

6.7.1. RPL Control Message Option Generic Format

RPL Control Message options all follow this format:

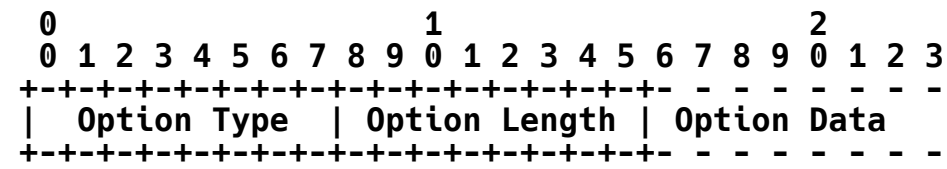


Figure 19: RPL Option Generic Format

Option Type: 8-bit identifier of the type of option. The Option Type values are assigned by IANA (see Section 20.4.)

Option Length: 8-bit unsigned integer, representing the length in octets of the option, not including the Option Type and Length fields.

Option Data: A variable length field that contains data specific to the option.

When processing a RPL message containing an option for which the Option Type value is not recognized by the receiver, the receiver MUST silently ignore the unrecognized option and continue to process the following option, correctly handling any remaining options in the message.

RPL message options may have alignment requirements. Following the convention in IPv6, options with alignment requirements are aligned in a packet such that multi-octet values within the Option Data field of each option fall on 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).

6.7.2. Pad1

The Pad1 option MAY be present in DIS, DIO, DAO, DAO-ACK, and CC messages, and its format is as follows:

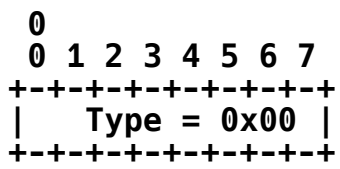


Figure 20: Format of the Pad1 Option

The Pad1 option is used to insert a single octet of padding into the message to enable options alignment. If more than one octet of padding is required, the PadN option should be used rather than multiple Pad1 options.

NOTE! The format of the Pad1 option is a special case -- it has neither Option Length nor Option Data fields.

6.7.3. PadN

The PadN option MAY be present in DIS, DIO, DAO, DAO-ACK, and CC messages, and its format is as follows:

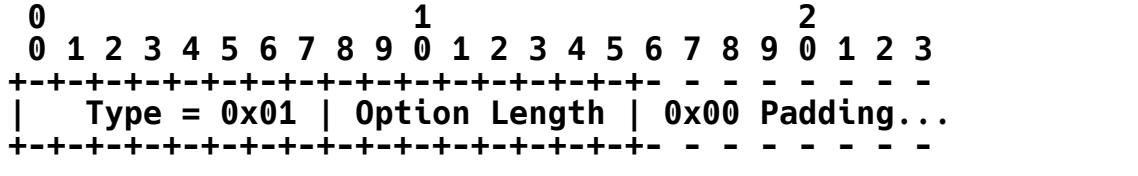


Figure 21: Format of the Pad N Option

The PadN option is used to insert two or more octets of padding into the message to enable options alignment. PadN option data MUST be ignored by the receiver.

Option Type: 0x01

Option Length: For N octets of padding, where 2 <= N <= 7, the Option Length field contains the value N-2. An Option Length of 0 indicates a total padding of 2 octets. An Option Length of 5 indicates a total padding of 7 octets, which is the maximum padding size allowed with the PadN option.

Option Data: For N (N > 1) octets of padding, the Option Data consists of N-2 zero-valued octets.

6.7.4. DAG Metric Container

The DAG Metric Container option MAY be present in DIO or DAO messages, and its format is as follows:

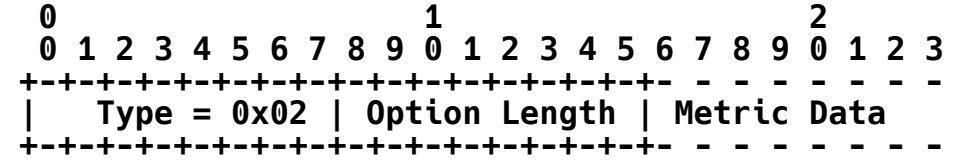


Figure 22: Format of the DAG Metric Container Option

The DAG Metric Container is used to report metrics along the DODAG. The DAG Metric Container may contain a number of discrete node, link, and aggregate path metrics and constraints specified in [RFC6551] as chosen by the implementer.

The DAG Metric Container MAY appear more than once in the same RPL control message, for example, to accommodate a use case where the Metric Data is longer than 256 bytes. More information is in [RFC6551].

The processing and propagation of the DAG Metric Container is governed by implementation specific policy functions.

Option Type: 0x02

Option Length: The Option Length field contains the length in octets of the Metric Data.

Metric Data: The order, content, and coding of the DAG Metric Container data is as specified in [RFC6551].

6.7.5. Route Information

The Route Information Option (RIO) MAY be present in DIO messages, and it carries the same information as the IPv6 Neighbor Discovery (ND) RIO as defined in [RFC4191]. The root of a DODAG is authoritative for setting that information and the information is unchanged as propagated down the DODAG. A RPL router may trivially transform it back into an ND option to advertise in its own RAs so a node attached to the RPL router will end up using the DODAG for which the root has the best preference for the destination of a packet. In addition to the existing ND semantics, it is possible for an Objective Function to use this information to favor a DODAG whose root is most preferred for a specific destination. The format of the option is modified slightly (Type, Length, Prefix) in order to be carried as a RPL option as follows:

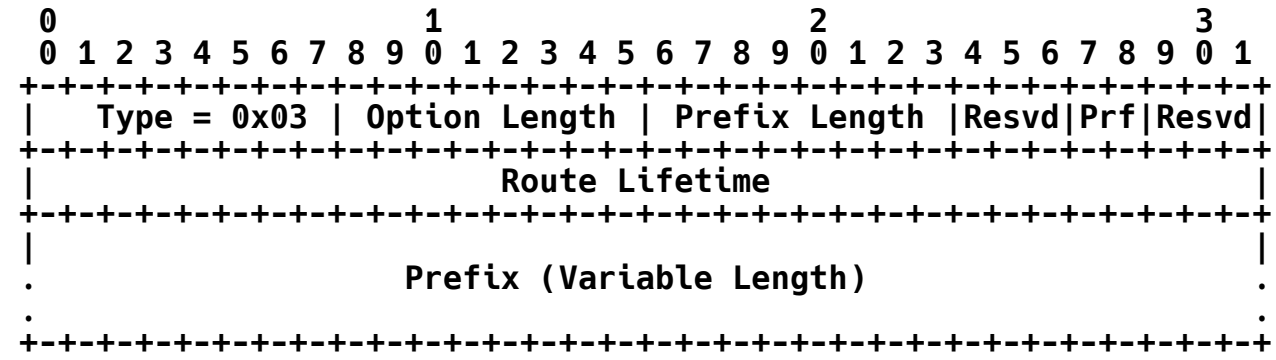


Figure 23: Format of the Route Information Option

The RIO is used to indicate that connectivity to the specified destination prefix is available from the DODAG root.

In the event that a RPL control message may need to specify connectivity to more than one destination, the RIO may be repeated.

[RFC4191] should be consulted as the authoritative reference with respect to the RIO. The field descriptions are transcribed here for convenience:

Option Type: 0x03

Option Length: Variable, length of the option in octets excluding the Type and Length fields. Note that this length is expressed in units of single octets, unlike in IPv6 ND.

Prefix Length: 8-bit unsigned integer. The number of leading bits in the prefix that are valid. The value ranges from 0 to 128. The Prefix field has the number of bytes inferred from the Option Length field, that must be at least the Prefix Length. Note that in RPL, this means that the Prefix field may have lengths other than 0, 8, or 16.

Prf: 2-bit signed integer. The Route Preference indicates whether to prefer the router associated with this prefix over others, when multiple identical prefixes (for different routers) have been received. If the Reserved (10) value is received, the RIO MUST be ignored. Per [RFC4191], the Reserved (10) value MUST NOT be sent. ([RFC4191] restricts the Preference to just three values to reinforce that it is not a metric.)

Resvd: Two 3-bit unused fields. They MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Route 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 route determination. A value of all one bits (0xFFFFFFFF) represents infinity.

Prefix: Variable-length field containing an IP address or a prefix of an IPv6 address. The Prefix Length field contains the number of valid leading bits in the prefix. The bits in the prefix after the prefix length (if any) are reserved and MUST be initialized to zero by the sender and ignored by the receiver. Note that in RPL, this field may have lengths other than 0, 8, or 16.

Unassigned bits of the RIO are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

6.7.6. DODAG Configuration

The DODAG Configuration option MAY be present in DIO messages, and its format is as follows:

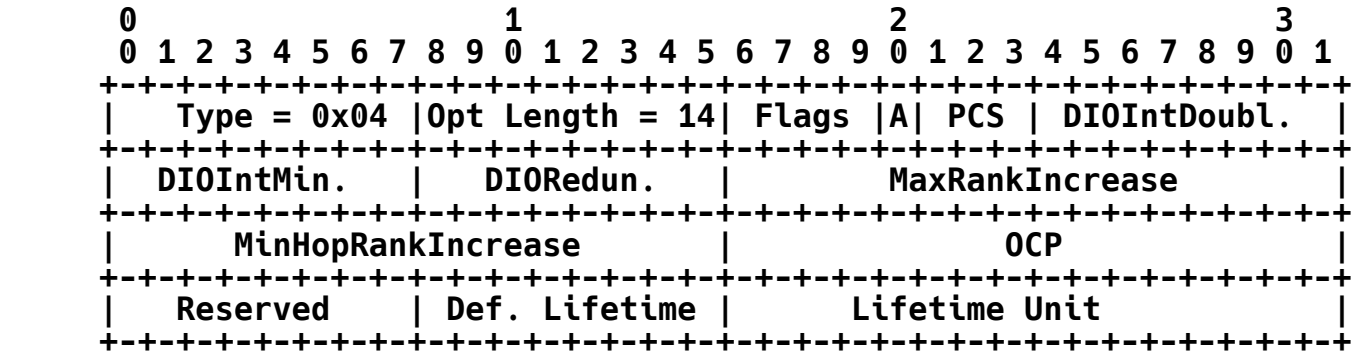


Figure 24: Format of the DODAG Configuration Option

The DODAG Configuration option is used to distribute configuration information for DODAG Operation through the DODAG.

The information communicated in this option is generally static and unchanging within the DODAG, therefore it is not necessary to include in every DIO. This information is configured at the DODAG root and distributed throughout the DODAG with the DODAG Configuration option. Nodes other than the DODAG root MUST NOT modify this information when propagating the DODAG Configuration option. This option MAY be included occasionally by the DODAG root (as determined by the DODAG root), and MUST be included in response to a unicast request, e.g. a unicast DODAG Information Solicitation (DIS) message.

Option Type: 0x04

Option Length: 14

Flags: The 4-bits remaining unused in the Flags field are reserved for flags. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Authentication Enabled (A): 1-bit flag describing the security mode of the network. The bit describes whether a node must authenticate with a key authority before joining the network as a router. If the DIO is not a secure DIO, the 'A' bit MUST be zero.

Path Control Size (PCS): 3-bit unsigned integer used to configure the number of bits that may be allocated to the Path Control field (see Section 9.9). Note that when PCS is consulted to determine the width of the Path Control field, a value of 1 is added, i.e., a PCS value of 0 results in 1 active bit in the Path Control field. The default value of PCS is DEFAULT_PATH_CONTROL_SIZE.

DIOIntervalDoublings: 8-bit unsigned integer used to configure Imax of the DIO Trickle timer (see Section 8.3.1). The default value of DIOIntervalDoublings is DEFAULT_DIO_INTERVAL_DOUBLINGS.

DIOIntervalMin: 8-bit unsigned integer used to configure Imin of the DIO Trickle timer (see Section 8.3.1). The default value of DIOIntervalMin is DEFAULT_DIO_INTERVAL_MIN.

DIORedundancyConstant: 8-bit unsigned integer used to configure k of the DIO Trickle timer (see Section 8.3.1). The default value of DIORedundancyConstant is DEFAULT_DIO_REDUNDANCY_CONSTANT.

MaxRankIncrease: 16-bit unsigned integer used to configure DAGMaxRankIncrease, the allowable increase in Rank in support of local repair. If DAGMaxRankIncrease is 0, then this mechanism is disabled.

MinHopRankIncrease: 16-bit unsigned integer used to configure MinHopRankIncrease as described in Section 3.5.1. The default value of MinHopRankInc is DEFAULT_MIN_HOP_RANK_INCREASE.

Objective Code Point (OCP): 16-bit unsigned integer. The OCP field identifies the OF and is managed by the IANA.

Reserved: 7-bit unused field. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Default Lifetime: 8-bit unsigned integer. This is the lifetime that is used as default for all RPL routes. It is expressed in units of Lifetime Units, e.g., the default lifetime in seconds is (Default Lifetime) * (Lifetime Unit).

Lifetime Unit: 16-bit unsigned integer. Provides the unit in seconds that is used to express route lifetimes in RPL. For very stable networks, it can be hours to days.

6.7.7. RPL Target

The RPL Target option MAY be present in DAO messages, and its format is as follows:

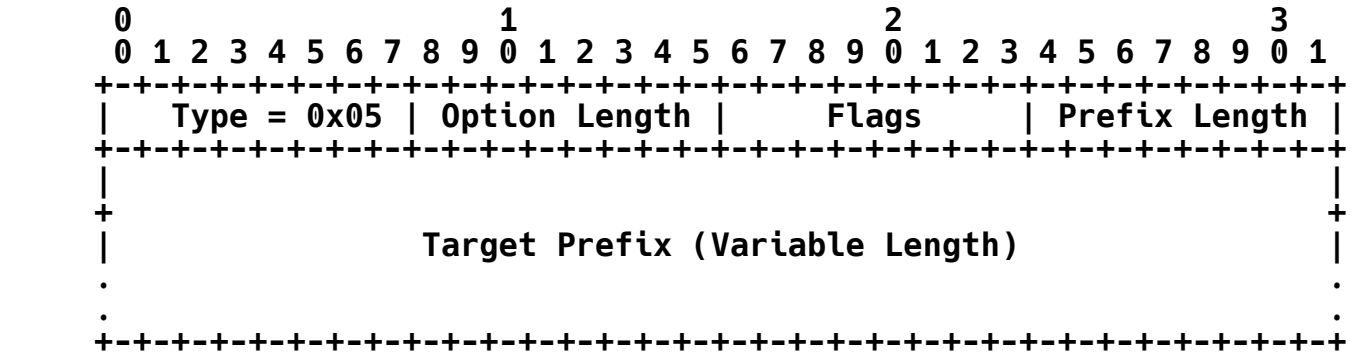


Figure 25: Format of the RPL Target Option

The RPL Target option is used to indicate a Target IPv6 address, prefix, or multicast group that is reachable or queried along the DODAG. In a DAO, the RPL Target option indicates reachability.

A RPL Target option MAY optionally be paired with a RPL Target Descriptor option (Figure 30) that qualifies the target.

A set of one or more Transit Information options (Section 6.7.8) MAY directly follow a set of one or more Target options in a DAO message (where each Target option MAY be paired with a RPL Target Descriptor option as above). The structure of the DAO message, detailing how Target options are used in conjunction with Transit Information options is further described in Section 9.4.

The RPL Target option may be repeated as necessary to indicate multiple targets.

Option Type: 0x05

Option Length: Variable, length of the option in octets excluding the Type and Length fields.

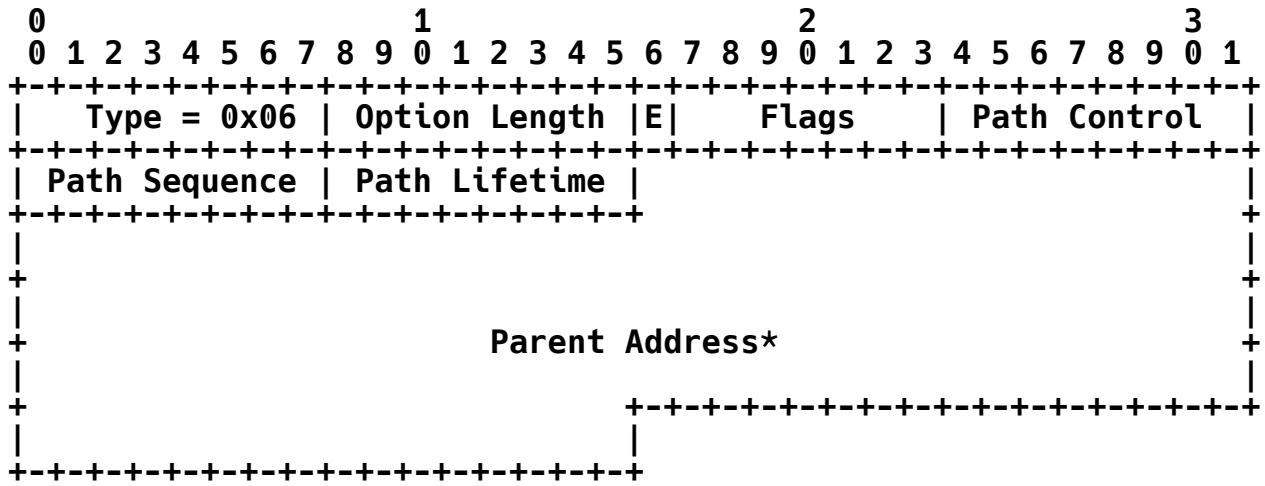
Flags: 8-bit unused field reserved for flags. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Prefix Length: 8-bit unsigned integer. Number of valid leading bits in the IPv6 Prefix.

Target Prefix: Variable-length field identifying an IPv6 destination address, prefix, or multicast group. The Prefix Length field contains the number of valid leading bits in the prefix. The bits in the prefix after the prefix length (if any) are reserved and MUST be set to zero on transmission and MUST be ignored on receipt.

6.7.8. Transit Information

The Transit Information option MAY be present in DAO messages, and its format is as follows:



The '*' denotes that the DODAG Parent Address subfield is not always present, as described below.

Figure 26: Format of the Transit Information Option

The Transit Information option is used for a node to indicate attributes for a path to one or more destinations. The destinations are indicated by one or more Target options that immediately precede the Transit Information option(s).

The Transit Information option can be used for a node to indicate its DODAG parents to an ancestor that is collecting DODAG routing information, typically, for the purpose of constructing source routes. In the Non-Storing mode of operation, this ancestor will be the DODAG root, and this option is carried by the DAO message. In the Storing mode of operation, the DODAG Parent Address subfield is not needed, since the DAO message is sent directly to the parent. The option length is used to determine whether or not the DODAG Parent Address subfield is present.

A non-storing node that has more than one DAO parent MAY include a Transit Information option for each DAO parent as part of the non-storing destination advertisement operation. The node may distribute the bits in the Path Control field among different groups of DAO parents in order to signal a preference among parents. That preference may influence the decision of the DODAG root when selecting among the alternate parents/paths for constructing Downward routes.

One or more Transit Information options MUST be preceded by one or more RPL Target options. In this manner, the RPL Target option indicates the child node, and the Transit Information option(s) enumerates the DODAG parents. The structure of the DAO message, further detailing how Target options are used in conjunction with Transit Information options, is further described in Section 9.4.

A typical non-storing node will use multiple Transit Information options, and it will send the DAO message thus formed directly to the root. A typical storing node will use one Transit Information option with no parent field and will send the DAO message thus formed, with additional adjustments, to Path Control as detailed later, to one or multiple parents.

For example, in a Non-Storing mode of operation let Tgt(T) denote a Target option for a Target T. Let Trnst(P) denote a Transit Information option that contains a parent address P. Consider the case of a non-storing Node N that advertises the self-owned targets N1 and N2 and has parents P1, P2, and P3. In that case, the DAO message would be expected to contain the sequence ((Tgt(N1), Tgt(N2)), (Trnst(P1), Trnst(P2), Trnst(P3))), such that the group of Target options {N1, N2} is described by the Transit Information options as having the parents {P1, P2, P3}. The non-storing node would then address that DAO message directly to the DODAG root and forward that DAO message through one of the DODAG parents: P1, P2, or P3.

Option Type: 0x06

Option Length: Variable, depending on whether or not the DODAG Parent Address subfield is present.

External (E): 1-bit flag. The 'E' flag is set to indicate that the parent router redistributes external targets into the RPL network. An external Target is a Target that has been learned through an alternate protocol. The external targets are listed in the Target options that immediately precede the Transit Information option. An external Target is not expected to support RPL messages and options.

Flags: The 7 bits remaining unused in the Flags field are reserved for flags. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Path Control: 8-bit bit field. The Path Control field limits the number of DAO parents to which a DAO message advertising connectivity to a specific destination may be sent, as well as providing some indication of relative preference. The limit provides some bound on overall DAO message fan-out in the LLN. The assignment and ordering of the bits in the Path Control also serves to communicate preference. Not all of these bits may be enabled as according to the PCS in the DODAG Configuration. The Path Control field is divided into four subfields that contain two bits each: PC1, PC2, PC3, and PC4, as illustrated in Figure 27. The subfields are ordered by preference, with PC1 being the most preferred and PC4 being the least preferred. Within a subfield, there is no order of preference. By grouping the parents (as in ECMP) and ordering them, the parents may be associated with specific bits in the Path Control field in a way that communicates preference.

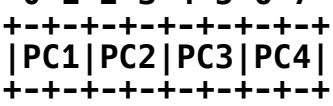


Figure 27: Path Control Preference Subfield Encoding

Path Sequence: 8-bit unsigned integer. When a RPL Target option is issued by the node that owns the Target prefix (i.e., in a DAO message), that node sets the Path Sequence and increments the Path Sequence each time it issues a RPL Target option with updated information.

Path Lifetime: 8-bit unsigned integer. The length of time in Lifetime Units (obtained from the Configuration option) that the prefix is valid for route determination. The period starts when a new Path Sequence is seen. A value of all one bits (0xFF) represents infinity. A value of all zero bits (0x00) indicates a loss of reachability. A DAO message that contains a Transit Information option with a Path Lifetime of 0x00 for a Target is referred as a No-Path (for that Target) in this document.

Parent Address (optional): IPv6 address of the DODAG parent of the node originally issuing the Transit Information option. This field may not be present, as according to the DODAG Mode of Operation (Storing or Non-Storing) and indicated by the Transit Information option length.

Unassigned bits of the Transit Information option are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

6.7.9. Solicited Information

The Solicited Information option MAY be present in DIS messages, and its format is as follows:

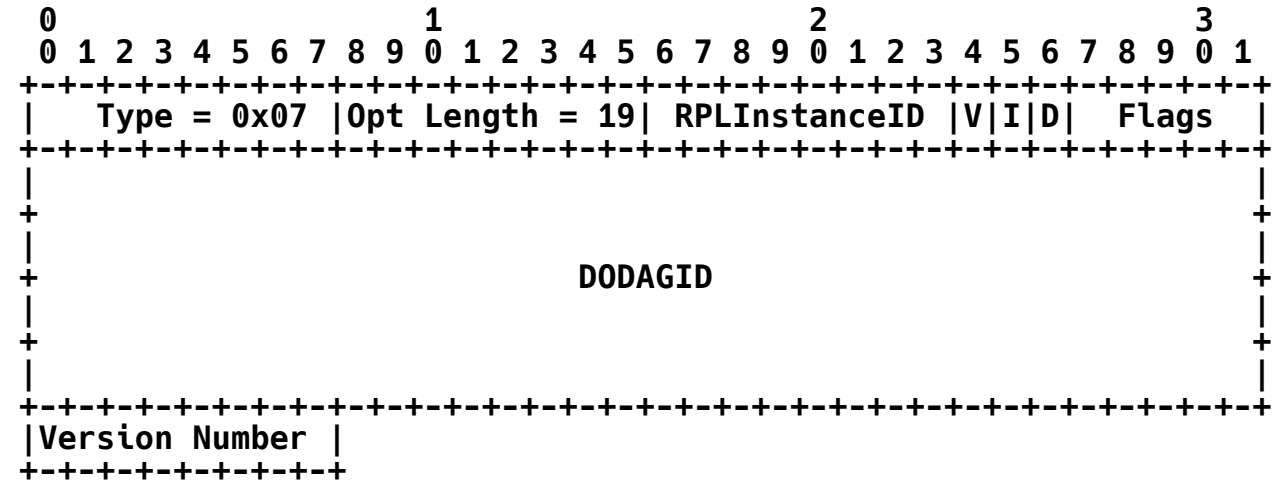


Figure 28: Format of the Solicited Information Option

The Solicited Information option is used for a node to request DIO messages from a subset of neighboring nodes. The Solicited Information option may specify a number of predicate criteria to be matched by a receiving node. This is used by the requester to limit the number of replies from "non-interesting" nodes. These predicates affect whether a node resets its DIO Trickle timer, as described in Section 8.3.

The Solicited Information option contains flags that indicate which predicates a node should check when deciding whether to reset its Trickle timer. A node resets its Trickle timer when all predicates are true. If a flag is set, then the RPL node MUST check the associated predicate. If a flag is cleared, then the RPL node MUST NOT check the associated predicate. (If a flag is cleared, the RPL node assumes that the associated predicate is true.)

Option Type: 0x07

Option Length: 19

V: The 'V' flag is the Version predicate. The Version predicate is true if the receiver's DODAGVersionNumber matches the requested Version Number. If the 'V' flag is cleared, then the Version field is not valid and the Version field MUST be set to zero on transmission and ignored upon receipt.

I: The 'I' flag is the InstanceID predicate. The InstanceID predicate is true when the RPL node's current RPLInstanceID matches the requested RPLInstanceID. If the 'I' flag is cleared, then the RPLInstanceID field is not valid and the RPLInstanceID field MUST be set to zero on transmission and ignored upon receipt.

D: The 'D' flag is the DODAGID predicate. The DODAGID predicate is true if the RPL node's parent set has the same DODAGID as the DODAGID field. If the 'D' flag is cleared, then the DODAGID field is not valid and the DODAGID field MUST be set to zero on transmission and ignored upon receipt.

Flags: The 5 bits remaining unused in the Flags field are reserved for flags. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Version Number: 8-bit unsigned integer containing the value of DODAGVersionNumber that is being solicited when valid.

RPLInstanceID: 8-bit unsigned integer containing the RPLInstanceID that is being solicited when valid.

DODAGID: 128-bit unsigned integer containing the DODAGID that is being solicited when valid.

Unassigned bits of the Solicited Information option are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

6.7.10. Prefix Information

The Prefix Information Option (PIO) MAY be present in DIO messages, and carries the information that is specified for the IPv6 ND Prefix Information option in [RFC4861], [RFC4862], and [RFC6275] for use by RPL nodes and IPv6 hosts. In particular, a RPL node may use this option for the purpose of Stateless Address Autoconfiguration (SLAAC) from a prefix advertised by a parent as specified in [RFC4862], and

advertise its own address as specified in [RFC6275]. The root of a DODAG is authoritative for setting that information. The information is propagated down the DODAG unchanged, with the exception that a RPL router may overwrite the Interface ID if the 'R' flag is set to indicate its full address in the PIO. The format of the option is modified (Type, Length, Prefix) in order to be carried as a RPL option as follows:

If the only desired effect of a received PIO in a DIO is to provide the global address of the parent node to the receiving node, then the sender resets the 'A' and 'L' bits and sets the 'R' bit. Upon receipt, the RPL will not autoconfigure an address or a connected route from the prefix [RFC4862]. As in all cases, when the 'L' bit is not set, the RPL node MAY include the prefix in PIOs it sends to its children.

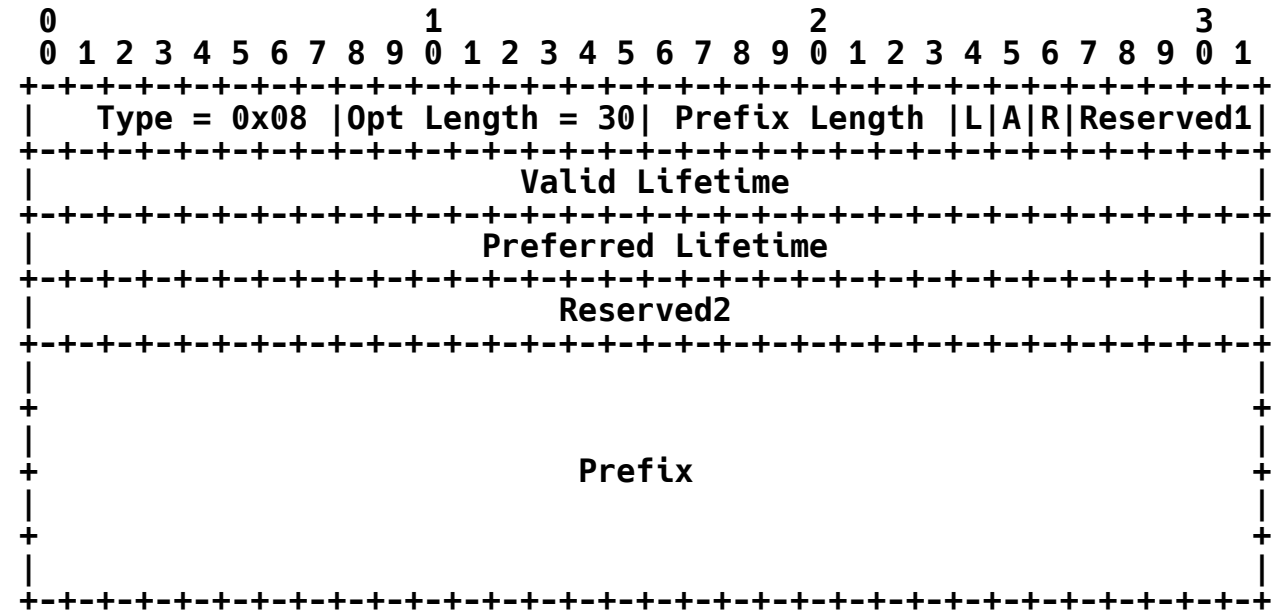


Figure 29: Format of the Prefix Information Option

The PIO may be used to distribute the prefix in use inside the DODAG, e.g., for address autoconfiguration.

[RFC4861] and [RFC6275] should be consulted as the authoritative reference with respect to the PIO. The field descriptions are transcribed here for convenience:

Option Type: 0x08

Option Length: 30. Note that this length is expressed in units of single octets, unlike in IPv6 ND.

Prefix Length: 8-bit unsigned integer. The number of leading bits in the Prefix field that are valid. The value ranges from 0 to 128. The Prefix Length field provides necessary information for on-link determination (when combined with the 'L' flag in the PIO). It also assists with address autoconfiguration as specified in [RFC4862], for which there may be more restrictions on the prefix length.

L: 1-bit on-link flag. When set, it indicates that this prefix can be used for on-link determination. When not set, the advertisement makes no statement about on-link or off-link properties of the prefix. In other words, if the 'L' flag is not set, a RPL node MUST NOT conclude that an address derived from the prefix is off-link. That is, it MUST NOT update a previous indication that the address is on-link. A RPL node acting as a router MUST NOT propagate a PIO with the 'L' flag set. A RPL node acting as a router MAY propagate a PIO with the 'L' flag not set.

A: 1-bit autonomous address-configuration flag. When set, it indicates that this prefix can be used for stateless address configuration as specified in [RFC4862]. When both protocols (ND RAs and RPL DIOs) are used to carry PIOs on the same link, it is possible to use either one for SLAAC by a RPL node. It is also possible to make either protocol ineligible for SLAAC operation by forcing the 'A' flag to 0 for PIOs carried in that protocol.

R: 1-bit router address flag. When set, it indicates that the Prefix field contains a complete IPv6 address assigned to the sending router that can be used as parent in a target option. The indicated prefix is the first prefix length bits of the Prefix field. The router IPv6 address has the same scope and conforms to the same lifetime values as the advertised prefix. This use of the Prefix field is compatible with its use in advertising the prefix itself, since Prefix Advertisement uses only the leading bits. Interpretation of this flag bit is thus independent of the processing required for the on-link (L) and autonomous address-configuration (A) flag bits.

Reserved1: 5-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 [RFC4862].

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 [RFC4862]. A value of all one bits (0xFFFFFFFF) represents infinity. See [RFC4862]. Note that the value of this field MUST NOT exceed the Valid Lifetime field to avoid preferring addresses that are no longer valid.

Reserved2: This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Prefix: An IPv6 address or a prefix of an IPv6 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. A non-storing node SHOULD refrain from advertising a prefix till it owns an address of that prefix, and then it SHOULD advertise its full address in this field, with the 'R' flag set. The children of a node that so advertises a full address with the 'R' flag set may then use that address to determine the content of the DODAG Parent Address subfield of the Transit Information option.

Unassigned bits of the PIO are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

6.7.11. RPL Target Descriptor

The RPL Target option MAY be immediately followed by one opaque descriptor that qualifies that specific target.

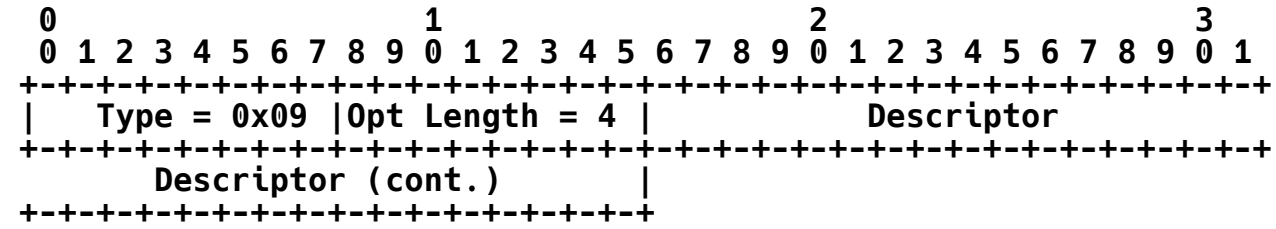


Figure 30: Format of the RPL Target Descriptor Option

The RPL Target Descriptor option is used to qualify a target, something that is sometimes called "tagging".

At most, there can be one descriptor per target. The descriptor is set by the node that injects the Target in the RPL network. It MUST be copied but not modified by routers that propagate the Target Up the DODAG in DAO messages.

Option Type: 0x09

Option Length: 4

Descriptor: 32-bit unsigned integer. Opaque.

7. Sequence Counters

This section describes the general scheme for bootstrap and operation of sequence counters in RPL, such as the DODAGVersionNumber in the DIO message, the DAOSequence in the DAO message, and the Path Sequence in the Transit Information option.

7.1. Sequence Counter Overview

This specification utilizes three different sequence numbers to validate the freshness and the synchronization of protocol information:

DODAGVersionNumber: This sequence counter is present in the DIO Base to indicate the Version of the DODAG being formed. The DODAGVersionNumber is monotonically incremented by the root each time the root decides to form a new Version of the DODAG in order to revalidate the integrity and allow a global repair to occur. The DODAGVersionNumber is propagated unchanged Down

the DODAG as routers join the new DODAG Version. The DODAGVersionNumber is globally significant in a DODAG and indicates the Version of the DODAG in which a router is operating. An older (lesser) value indicates that the originating router has not migrated to the new DODAG Version and cannot be used as a parent once the receiving node has migrated to the newer DODAG Version.

DAOSequence: This sequence counter is present in the DAO Base to correlate a DAO message and a DAO ACK message. The DAOSequence number is locally significant to the node that issues a DAO message for its own consumption to detect the loss of a DAO message and enable retries.

Path Sequence: This sequence counter is present in the Transit Information option in a DAO message. The purpose of this counter is to differentiate a movement where a newer route supersedes a stale one from a route redundancy scenario where multiple routes exist in parallel for the same target. The Path Sequence is globally significant in a DODAG and indicates the freshness of the route to the associated target. An older (lesser) value received from an originating router indicates that the originating router holds stale routing states and the originating router should not be considered anymore as a potential next hop for the target. The Path Sequence is computed by the node that advertises the target, that is the Target itself or a router that advertises a Target on behalf of a host, and is unchanged as the DAO content is propagated towards the root by parent routers. If a host does not pass a counter to its router, then the router is in charge of computing the Path Sequence on behalf of the host and the host can only register to one router for that purpose. If a DAO message containing the same Target is issued to multiple parents at a given point in time for the purpose of route redundancy, then the Path Sequence is the same in all the DAO messages for that same target.

7.2. Sequence Counter Operation

RPL sequence counters are subdivided in a 'lollipop' fashion [Perlman83], where the values from 128 and greater are used as a linear sequence to indicate a restart and bootstrap the counter, and the values less than or equal to 127 used as a circular sequence number space of size 128 as in [RFC1982]. Consideration is given to the mode of operation when transitioning from the linear region to the circular region. Finally, when operating in the circular region, if sequence numbers are detected to be too far apart, then they are not comparable, as detailed below.

A window of comparison, SEQUENCE_WINDOW = 16, is configured based on a value of 2^N, where N is defined to be 4 in this specification.

For a given sequence counter:

1. The sequence counter SHOULD be initialized to an implementation defined value, which is 128 or greater prior to use. A recommended value is 240 (256 - SEQUENCE_WINDOW).

2. When a sequence counter increment would cause the sequence counter to increment beyond its maximum value, the sequence counter MUST wrap back to zero. When incrementing a sequence counter greater than or equal to 128, the maximum value is 255. When incrementing a sequence counter less than 128, the maximum value is 127.

3. When comparing two sequence counters, the following rules MUST be applied:

    1. When a first sequence counter A is in the interval [128..255] and a second sequence counter B is in [0..127]:

        1. If (256 + B - A) is less than or equal to SEQUENCE_WINDOW, then B is greater than A, A is less than B, and the two are not equal.

        2. If (256 + B - A) is greater than SEQUENCE_WINDOW, then A is greater than B, B is less than A, and the two are not equal.

        For example, if A is 240, and B is 5, then (256 + 5 - 240) is 21. 21 is greater than SEQUENCE_WINDOW (16); thus, 240 is greater than 5. As another example, if A is 250 and B is 5, then (256 + 5 - 250) is 11. 11 is less than SEQUENCE_WINDOW (16); thus, 250 is less than 5.

    2. In the case where both sequence counters to be compared are less than or equal to 127, and in the case where both sequence counters to be compared are greater than or equal to 128:

        1. If the absolute magnitude of difference between the two sequence counters is less than or equal to SEQUENCE_WINDOW, then a comparison as described in [RFC1982] is used to determine the relationships greater than, less than, and equal.

2. If the absolute magnitude of difference of the two sequence counters is greater than SEQUENCE_WINDOW, then a desynchronization has occurred and the two sequence numbers are not comparable.

4. If two sequence numbers are determined not to be comparable, i.e., the results of the comparison are not defined, then a node should consider the comparison as if it has evaluated in such a way so as to give precedence to the sequence number that has most recently been observed to increment. Failing this, the node should consider the comparison as if it has evaluated in such a way so as to minimize the resulting changes to its own state.

8. Upward Routes

This section describes how RPL discovers and maintains Upward routes. It describes the use of DODAG Information Objects (DIOs), the messages used to discover and maintain these routes. It specifies how RPL generates and responds to DIOs. It also describes DODAG Information Solicitation (DIS) messages, which are used to trigger DIO transmissions.

As mentioned in Section 3.2.8, nodes that decide to join a DODAG MUST provision at least one DODAG parent as a default route for the associated instance. This default route enables a packet to be forwarded Upward until it eventually hits a common ancestor from which it will be routed Downward to the destination. If the destination is not in the DODAG, then the DODAG root may be able to forward the packet using connectivity to the outside of the DODAG; if it cannot forward the packet outside, then the DODAG root has to drop it.

A DIO message can also transport explicit routing information:

DODAGID: The DODAGID is a Global or Unique Local IPv6 address of the root. A node that joins a DODAG SHOULD provision a host route via a DODAG parent to the address used by the root as the DODAGID.

RIO Prefix: The root MAY place one or more Route Information options in a DIO message. The RIO is used to advertise an external route that is reachable via the root, associated with a preference, as presented in Section 6.7.5, which incorporates the RIO from [RFC4191]. It is interpreted as a capability of the root as opposed to a routing advertisement, and it MUST NOT be redistributed in another routing protocol though it SHOULD be used by an ingress RPL router to select a DODAG when a packet is injected in a RPL domain from a node attached to that

RPL router. An Objective Function MAY use the routes advertised in RIO or the preference for those routes in order to favor a DODAG versus another one for the same instance.

8.1. DIO Base Rules

1. For the following DIO Base fields, a node that is not a DODAG root MUST advertise the same values as its preferred DODAG parent (defined in Section 8.2.1). In this way, these values will propagate Down the DODAG unchanged and advertised by every node that has a route to that DODAG root. These fields are as follows:
   1. Grounded (G)
   2. Mode of Operation (MOP)
   3. DAGPreference (Prf)
   4. Version
   5. RPLInstanceID
   6. DODAGID

2. A node MAY update the following fields at each hop:
   1. Rank
   2. DTSN

3. The DODAGID field each root sets MUST be unique within the RPL Instance and MUST be a routable IPv6 address belonging to the root.

8.2. Upward Route Discovery and Maintenance

Upward route discovery allows a node to join a DODAG by discovering neighbors that are members of the DODAG of interest and identifying a set of parents. The exact policies for selecting neighbors and parents is implementation dependent and driven by the OF. This section specifies the set of rules those policies must follow for interoperability.

8.2.1. Neighbors and Parents within a DODAG Version

RPL's Upward route discovery algorithms and processing are in terms of three logical sets of link-local nodes. First, the candidate neighbor set is a subset of the nodes that can be reached via link-local multicast. The selection of this set is implementation and OF dependent. Second, the parent set is a restricted subset of the candidate neighbor set. Finally, the preferred parent is a member of the parent set that is the preferred next hop in Upward routes. Conceptually, the preferred parent is a single parent; although, it may be a set of multiple parents if those parents are equally preferred and have identical Rank.

More precisely:

1. The DODAG parent set MUST be a subset of the candidate neighbor set.

2. A DODAG root MUST have a DODAG parent set of size zero.

3. A node that is not a DODAG root MAY maintain a DODAG parent set of size greater than or equal to one.

4. A node's preferred DODAG parent MUST be a member of its DODAG parent set.

5. A node's Rank MUST be greater than all elements of its DODAG parent set.

6. When Neighbor Unreachability Detection (NUD) [RFC4861], or an equivalent mechanism, determines that a neighbor is no longer reachable, a RPL node MUST NOT consider this node in the candidate neighbor set when calculating and advertising routes until it determines that it is again reachable. Routes through an unreachable neighbor MUST be removed from the routing table.

These rules ensure that there is a consistent partial order on nodes within the DODAG. As long as node Ranks do not change, following the above rules ensures that every node's route to a DODAG root is loop-free, as Rank decreases on each hop to the root.

The OF can guide candidate neighbor set and parent set selection, as discussed in [RFC6552].

8.2.2. Neighbors and Parents across DODAG Versions

The above rules govern a single DODAG Version. The rules in this section define how RPL operates when there are multiple DODAG Versions.

8.2.2.1. DODAG Version

1. The tuple (RPLInstanceID, DODAGID, DODAGVersionNumber) uniquely defines a DODAG Version. Every element of a node's DODAG parent set, as conveyed by the last heard DIO message from each DODAG parent, MUST belong to the same DODAG Version. Elements of a node's candidate neighbor set MAY belong to different DODAG Versions.

2. A node is a member of a DODAG Version if every element of its DODAG parent set belongs to that DODAG Version, or if that node is the root of the corresponding DODAG.

3. A node MUST NOT send DIOs for DODAG Versions of which it is not a member.

4. DODAG roots MAY increment the DODAGVersionNumber that they advertise and thus move to a new DODAG Version. When a DODAG root increments its DODAGVersionNumber, it MUST follow the conventions of Serial Number Arithmetic as described in Section 7. Events triggering the increment of the DODAGVersionNumber are described later in this section and in Section 18.

5. Within a given DODAG, a node that is a not a root MUST NOT advertise a DODAGVersionNumber higher than the highest DODAGVersionNumber it has heard. Higher is defined as the greater-than operator in Section 7.

6. Once a node has advertised a DODAG Version by sending a DIO, it MUST NOT be a member of a previous DODAG Version of the same DODAG (i.e., with the same RPLInstanceID, the same DODAGID, and a lower DODAGVersionNumber). Lower is defined as the less-than operator in Section 7.

When the DODAG parent set becomes empty on a node that is not a root, (i.e., the last parent has been removed, causing the node no longer to be associated with that DODAG), then the DODAG information should not be suppressed until after the expiration of an implementation-specific local timer. During the interval prior to suppression of the "old" DODAG state, the node will be able to observe if the DODAGVersionNumber has been incremented should any new parents appear. This will help protect against the possibility of loops that may occur if that node were to inadvertently rejoin the old DODAG Version in its own prior sub-DODAG.

As the DODAGVersionNumber is incremented, a new DODAG Version spreads outward from the DODAG root. A parent that advertises the new DODAGVersionNumber cannot belong to the sub-DODAG of a node advertising an older DODAGVersionNumber. Therefore, a node can safely add a parent of any Rank with a newer DODAGVersionNumber without forming a loop.

For example, suppose that a node has left a DODAG with DODAGVersionNumber N. Suppose that a node had a sub-DODAG and did attempt to poison that sub-DODAG by advertising a Rank of INFINITE_RANK, but those advertisements may have become lost in the

LLN. Then, if the node did observe a candidate neighbor advertising a position in that original DODAG at DODAGVersionNumber N, that candidate neighbor could possibly have been in the node's former sub-DODAG, and there is a possible case where adding that candidate neighbor as a parent could cause a loop. In this case, if that candidate neighbor is observed to advertise a DODAGVersionNumber N+1, then that candidate neighbor is certain to be safe, since it is certain not to be in that original node's sub-DODAG, as it has been able to increment the DODAGVersionNumber by hearing from the DODAG root while that original node was detached. For this reason, it is useful for the detached node to remember the original DODAG information, including the DODAGVersionNumber N.

Exactly when a DODAG root increments the DODAGVersionNumber is implementation dependent and out of scope for this specification. Examples include incrementing the DODAGVersionNumber periodically, upon administrative intervention, or on application-level detection of lost connectivity or DODAG inefficiency.

After a node transitions to and advertises a new DODAG Version, the rules above make it unable to advertise the previous DODAG Version (prior DODAGVersionNumber) once it has committed to advertising the new DODAG Version.

8.2.2.2. DODAG Roots

1. A DODAG root without possibility to satisfy the application-defined goal MUST NOT set the Grounded bit.

2. A DODAG root MUST advertise a Rank of ROOT_RANK.

3. A node whose DODAG parent set is empty MAY become the DODAG root of a floating DODAG. It MAY also set its DAGPreference such that it is less preferred.

In a deployment that uses non-LLN links to federate a number of LLN roots, it is possible to run RPL over those non-RPL links and use one router as a "backbone root". The backbone root is the virtual root of the DODAG and exposes a Rank of BASE_RANK over the backbone. All the LLN roots that are parented to that backbone root, including the backbone root if it also serves as the LLN root itself, expose a Rank of ROOT_RANK to the LLN. These virtual roots are part of the same DODAG and advertise the same DODAGID. They coordinate DODAGVersionNumbers and other DODAG parameters with the virtual root over the backbone. The method of coordination is out of scope for this specification (to be defined in future companion specifications).

8.2.2.3. DODAG Selection

The Objective Function and the set of advertised routing metrics and constraints of a DAG determine how a node selects its neighbor set, parent set, and preferred parents. This selection implicitly also determines the DODAG within a DAG. Such selection can include administrative preference (Prf) as well as metrics or other considerations.

If a node has the option to join a more preferred DODAG while still meeting other optimization objectives, then the node will generally seek to join the more preferred DODAG as determined by the OF. All else being equal, it is left to the implementation to determine which DODAG is most preferred (since, as a reminder, a node must only join one DODAG per RPL Instance).

8.2.2.4. Rank and Movement within a DODAG Version

1. A node MUST NOT advertise a Rank less than or equal to any member of its parent set within the DODAG Version.

2. A node MAY advertise a Rank lower than its prior advertisement within the DODAG Version.

3. Let L be the lowest Rank within a DODAG Version that a given node has advertised. Within the same DODAG Version, that node MUST NOT advertise an effective Rank higher than L + DAGMaxRankIncrease. INFINITE_RANK is an exception to this rule: a node MAY advertise an INFINITE_RANK within a DODAG Version without restriction. If a node's Rank were to be higher than allowed by L + DAGMaxRankIncrease, when it advertises Rank, it MUST advertise its Rank as INFINITE_RANK.

4. A node MAY, at any time, choose to join a different DODAG within a RPL Instance. Such a join has no Rank restrictions, unless that different DODAG is a DODAG Version of which this node has previously been a member; in which case, the rule of the previous bullet (3) must be observed. Until a node transmits a DIO indicating its new DODAG membership, it MUST forward packets along the previous DODAG.

5. A node MAY, at any time after hearing the next DODAGVersionNumber advertised from suitable DODAG parents, choose to migrate to the next DODAG Version within the DODAG.

Conceptually, an implementation is maintaining a DODAG parent set within the DODAG Version. Movement entails changes to the DODAG parent set. Moving Up does not present the risk to create a loop but moving Down might, so that operation is subject to additional constraints.

When a node migrates to the next DODAG Version, the DODAG parent set needs to be rebuilt for the new Version. An implementation could defer to migrate for some reasonable amount of time, to see if some other neighbors with potentially better metrics but higher Rank announce themselves. Similarly, when a node jumps into a new DODAG, it needs to construct a new DODAG parent set for this new DODAG.

If a node needs to move Down a DODAG that it is attached to, increasing its Rank, then it MAY poison its routes and delay before moving as described in Section 8.2.2.5.

A node is allowed to join any DODAG Version that it has never been a prior member of without any restrictions, but if the node has been a prior member of the DODAG Version, then it must continue to observe the rule that it may not advertise a Rank higher than L+DAGMaxRankIncrease at any point in the life of the DODAG Version. This rule must be observed so as not to create a loophole that would allow the node to effectively increment its Rank all the way to INFINITE_RANK, which may have impact on other nodes and create a resource-wasting count-to-infinity scenario.

8.2.2.5. Poisoning

1. A node poisons routes by advertising a Rank of INFINITE_RANK.

2. A node MUST NOT have any nodes with a Rank of INFINITE_RANK in its parent set.

Although an implementation may advertise INFINITE_RANK for the purposes of poisoning, doing so is not the same as setting Rank to INFINITE_RANK. For example, a node may continue to send data packets whose RPL Packet Information includes a Rank that is not INFINITE_RANK, yet still advertise INFINITE_RANK in its DIOs.

When a (former) parent is observed to advertise a Rank of INFINITE_RANK, that (former) parent has detached from the DODAG and is no longer able to act as a parent, nor is there any way that another node may be considered to have a Rank greater-than INFINITE_RANK. Therefore, that (former) parent cannot act as a parent any longer and is removed from the parent set.

8.2.2.6. Detaching

1. A node unable to stay connected to a DODAG within a given DODAG Version, i.e., that cannot retain non-empty parent set without violating the rules of this specification, MAY detach from this DODAG Version. A node that detaches becomes the root of its own floating DODAG and SHOULD immediately advertise this new situation in a DIO as an alternate to poisoning.

8.2.2.7. Following a Parent

1. If a node receives a DIO from one of its DODAG parents, indicating that the parent has left the DODAG, that node SHOULD stay in its current DODAG through an alternative DODAG parent, if possible. It MAY follow the leaving parent.

A DODAG parent may have moved, migrated to the next DODAG Version, or jumped to a different DODAG. A node ought to give some preference to remaining in the current DODAG, if possible via an alternate parent, but ought to follow the parent if there are no other options.

8.2.3. DIO Message Communication

When a DIO message is received, the receiving node must first determine whether or not the DIO message should be accepted for further processing, and subsequently present the DIO message for further processing if eligible.

1. If the DIO message is malformed, then the DIO message is not eligible for further processing and a node MUST silently discard it. (See Section 18 for error logging).

2. If the sender of the DIO message is a member of the candidate neighbor set and the DIO message is not malformed, the node MUST process the DIO.

8.2.3.1. DIO Message Processing

As DIO messages are received from candidate neighbors, the neighbors may be promoted to DODAG parents by following the rules of DODAG discovery as described in Section 8.2. When a node places a neighbor into the DODAG parent set, the node becomes attached to the DODAG through the new DODAG parent node.

The most preferred parent should be used to restrict which other nodes may become DODAG parents. Some nodes in the DODAG parent set may be of a Rank less than or equal to the most preferred DODAG parent. (This case may occur, for example, if an energy-constrained device is at a lesser Rank but should be avoided per an optimization objective, resulting in a more preferred parent at a greater Rank.)

8.3. DIO Transmission

RPL nodes transmit DIOs using a Trickle timer [RFC6206]. A DIO from a sender with a lesser DAGRank that causes no changes to the recipient's parent set, preferred parent, or Rank SHOULD be considered consistent with respect to the Trickle timer.

The following packets and events MUST be considered inconsistencies with respect to the Trickle timer, and cause the Trickle timer to reset:

- When a node detects an inconsistency when forwarding a packet, as detailed in Section 11.2.
- When a node receives a multicast DIS message without a Solicited Information option, unless a DIS flag restricts this behavior.
- When a node receives a multicast DIS with a Solicited Information option and the node matches all of the predicates in the Solicited Information option, unless a DIS flag restricts this behavior.
- When a node joins a new DODAG Version (e.g., by updating its DODAGVersionNumber, joining a new RPL Instance, etc.).

Note that this list is not exhaustive, and an implementation MAY consider other messages or events to be inconsistencies.

A node SHOULD NOT reset its DIO Trickle timer in response to unicast DIS messages. When a node receives a unicast DIS without a Solicited Information option, it MUST unicast a DIO to the sender in response. This DIO MUST include a DODAG Configuration option. When a node receives a unicast DIS message with a Solicited Information option and matches the predicates of that Solicited Information option, it MUST unicast a DIO to the sender in response. This unicast DIO MUST include a DODAG Configuration option. Thus, a node MAY transmit a unicast DIS message to a potential DODAG parent in order to probe for DODAG Configuration and other parameters.

8.3.1. Trickle Parameters

The configuration parameters of the Trickle timer are specified as follows:

Imin: learned from the DIO message as (2^DIOIntervalMin) ms. The default value of DIOIntervalMin is DEFAULT_DIO_INTERVAL_MIN.

Imax: learned from the DIO message as DIOIntervalDoublings. The default value of DIOIntervalDoublings is DEFAULT_DIO_INTERVAL_DOUBLINGS.

k: learned from the DIO message as DIORedundancyConstant. The default value of DIORedundancyConstant is DEFAULT_DIO_REDUNDANCY_CONSTANT. In RPL, when k has the value of 0x00, this is to be treated as a redundancy constant of infinity in RPL, i.e., Trickle never suppresses messages.

8.4. DODAG Selection

The DODAG selection is implementation and OF dependent. In order to limit erratic movements, and all metrics being equal, nodes SHOULD keep their previous selection. Also, nodes SHOULD provide a means to filter out a parent whose availability is detected as fluctuating, at least when more stable choices are available.

When connection to a grounded DODAG is not possible or preferable for security or other reasons, scattered DODAGs MAY aggregate as much as possible into larger DODAGs in order to allow connectivity within the LLN.

A node SHOULD verify that bidirectional connectivity and adequate link quality is available with a candidate neighbor before it considers that candidate as a DODAG parent.

8.5. Operation as a Leaf Node

In some cases, a RPL node may attach to a DODAG as a leaf node only. One example of such a case is when a node does not understand or does not support (policy) the RPL Instance's OF or advertised metric/constraint. As specified in Section 18.6, related to policy function, the node may either join the DODAG as a leaf node or may not join the DODAG. As mentioned in Section 18.5, it is then recommended to log a fault.

A leaf node does not extend DODAG connectivity; however, in some cases, the leaf node may still need to transmit DIOs on occasion, in particular, when the leaf node may not have always been acting as a leaf node and an inconsistency is detected.

A node operating as a leaf node must obey the following rules:

1. It MUST NOT transmit DIOs containing the DAG Metric Container.

2. Its DIOs MUST advertise a DAGRank of INFINITE_RANK.

3. It MAY suppress DIO transmission, unless the DIO transmission has been triggered due to detection of inconsistency when a packet is being forwarded or in response to a unicast DIS message, in which case the DIO transmission MUST NOT be suppressed.

4. It MAY transmit unicast DAOs as described in Section 9.2.

5. It MAY transmit multicast DAOs to the '1 hop' neighborhood as described in Section 9.10.

A particular case that requires a leaf node to send a DIO is if that leaf node was a prior member of another DODAG and another node forwards a message assuming the old topology, triggering an inconsistency. The leaf node needs to transmit a DIO in order to repair the inconsistency. Note that due to the lossy nature of LLNs, even though the leaf node may have optimistically poisoned its routes by advertising a Rank of INFINITE_RANK in the old DODAG prior to becoming a leaf node, that advertisement may have become lost and a leaf node must be capable to send a DIO later in order to repair the inconsistency.

In the general case, the leaf node MUST NOT advertise itself as a router (i.e., send DIOs).

8.6. Administrative Rank

In some cases, it might be beneficial to adjust the Rank advertised by a node beyond that computed by the OF based on some implementation-specific policy and properties of the node. For example, a node that has a limited battery should be a leaf unless there is no other choice, and may then augment the Rank computation specified by the OF in order to expose an exaggerated Rank.

9. Downward Routes

This section describes how RPL discovers and maintains Downward routes. RPL constructs and maintains Downward routes with Destination Advertisement Object (DAO) messages. Downward routes support P2MP flows, from the DODAG roots toward the leaves. Downward routes also support P2P flows: P2P messages can flow toward a DODAG root (or a common ancestor) through an Upward route, then away from the DODAG root to a destination through a Downward route.

This specification describes the two modes a RPL Instance may choose from for maintaining Downward routes. In the first mode, called "Storing", nodes store Downward routing tables for their sub-DODAG. Each hop on a Downward route in a storing network examines its routing table to decide on the next hop. In the second mode, called "Non-Storing", nodes do not store Downward routing tables. Downward packets are routed with source routes populated by a DODAG root [RFC6554].

RPL allows a simple one-hop P2P optimization for both storing and non-storing networks. A node may send a P2P packet destined to a one-hop neighbor directly to that node.

9.1. Destination Advertisement Parents

To establish Downward routes, RPL nodes send DAO messages Upward. The next-hop destinations of these DAO messages are called "DAO parents". The collection of a node's DAO parents is called the "DAO parent set".

1. A node MAY send DAO messages using the all-RPL-nodes multicast address, which is an optimization to provision one-hop routing. The 'K' bit MUST be cleared on transmission of the multicast DAO.

2. A node's DAO parent set MUST be a subset of its DODAG parent set.

3. In Storing mode operation, a node MUST NOT address unicast DAO messages to nodes that are not DAO parents.

4. In Storing mode operation, the IPv6 source and destination addresses of a DAO message MUST be link-local addresses.

5. In Non-Storing mode operation, a node MUST NOT address unicast DAO messages to nodes that are not DODAG roots.

6. In Non-Storing mode operation, the IPv6 source and destination addresses of a DAO message MUST be a unique-local or a global address.

The selection of DAO parents is implementation and Objective Function specific.

9.2. Downward Route Discovery and Maintenance

Destination Advertisement may be configured to be entirely disabled, or operate in either a Storing or Non-Storing mode, as reported in the MOP in the DIO message.

1. All nodes who join a DODAG MUST abide by the MOP setting from the root. Nodes that do not have the capability to fully participate as a router, e.g., that do not match the advertised MOP, MAY join the DODAG as a leaf.

2. If the MOP is 0, indicating no Downward routing, nodes MUST NOT transmit DAO messages and MAY ignore DAO messages.

3. In Non-Storing mode, the DODAG root SHOULD store source routing table entries for destinations learned from DAOs. The DODAG root MUST be able to generate source routes for those destinations learned from DAOs that were stored.

4. In Storing mode, all non-root, non-leaf nodes MUST store routing table entries for destinations learned from DAOs.

A DODAG can have one of several possible modes of operation, as defined by the MOP field. Either it does not support Downward routes, it supports Downward routes through source routing from DODAG roots, or it supports Downward routes through in-network routing tables.

When Downward routes are supported through source routing from DODAG roots, it is generally expected that the DODAG root has stored the source routing information learned from DAOs in order to construct the source routes. If the DODAG root fails to store some information, then some destinations may be unreachable.

When Downward routes are supported through in-network routing tables, the multicast operation defined in this specification may or may not be supported, also as indicated by the MOP field.

When Downward routes are supported through in-network routing tables, as described in this specification, it is expected that nodes acting as routers have been provisioned sufficiently to hold the required routing table state. If a node acting as a router is unable to hold the full routing table state then the routing state is not complete,

messages may be dropped as a consequence, and a fault may be logged (Section 18.5). Future extensions to RPL may elaborate on refined actions/behaviors to manage this case.

As of the writing of this specification, RPL does not support mixed-mode operation, where some nodes source route and other store routing tables: future extensions to RPL may support this mode of operation.

9.2.1. Maintenance of Path Sequence

For each Target that is associated with (owned by) a node, that node is responsible to emit DAO messages in order to provision the Downward routes. The Target+Transit information contained in those DAO messages subsequently propagates Up the DODAG. The Path Sequence counter in the Transit information option is used to indicate freshness and update stale Downward routing information as described in Section 7.

For a Target that is associated with (owned by) a node, that node MUST increment the Path Sequence counter, and generate a new DAO message, when:

1. the Path Lifetime is to be updated (e.g., a refresh or a no-Path).

2. the DODAG Parent Address subfield list is to be changed.

For a Target that is associated with (owned by) a node, that node MAY increment the Path Sequence counter, and generate a new DAO message, on occasion in order to refresh the Downward routing information. In Storing mode, the node generates such a DAO to each of its DAO parents in order to enable multipath. All DAOs generated at the same time for the same Target MUST be sent with the same Path Sequence in the Transit Information.

9.2.2. Generation of DAO Messages

A node might send DAO messages when it receives DAO messages, as a result of changes in its DAO parent set, or in response to another event such as the expiry of a related prefix lifetime. In the case of receiving DAOs, it matters whether the DAO message is "new" or contains new information. In Non-Storing mode, every DAO message a node receives is "new". In Storing mode, a DAO message is "new" if it satisfies any of these criteria for a contained Target:

1. it has a newer Path Sequence number,

2. it has additional Path Control bits, or

3. it is a No-Path DAO message that removes the last Downward route to a prefix.

A node that receives a DAO message from its sub-DODAG MAY suppress scheduling a DAO message transmission if that DAO message is not new.

9.3. DAO Base Rules

1. If a node sends a DAO message with newer or different information than the prior DAO message transmission, it MUST increment the DAOSequence field by at least one. A DAO message transmission that is identical to the prior DAO message transmission MAY increment the DAOSequence field.

2. The RPLInstanceID and DODAGID fields of a DAO message MUST be the same value as the members of the node's parent set and the DIOs it transmits.

3. A node MAY set the 'K' flag in a unicast DAO message to solicit a unicast DAO-ACK in response in order to confirm the attempt.

4. A node receiving a unicast DAO message with the 'K' flag set SHOULD respond with a DAO-ACK. A node receiving a DAO message without the 'K' flag set MAY respond with a DAO-ACK, especially to report an error condition.

5. A node that sets the 'K' flag in a unicast DAO message but does not receive a DAO-ACK in response MAY reschedule the DAO message transmission for another attempt, up until an implementation-specific number of retries.

6. Nodes SHOULD ignore DAOs without newer sequence numbers and MUST NOT process them further.

Unlike the Version field of a DIO, which is incremented only by a DODAG root and repeated unchanged by other nodes, DAOSequence values are unique to each node. The sequence number space for unicast and multicast DAO messages can be either the same or distinct. It is RECOMMENDED to use the same sequence number space.

9.4. Structure of DAO Messages

DAOs follow a common structure in both storing and non-storing networks. In the most general form, a DAO message may include several groups of options, where each group consists of one or more Target options followed by one or more Transit Information options.

The entire group of Transit Information options applies to the entire group of Target options. Later sections describe further details for each mode of operation.

1. RPL nodes MUST include one or more RPL Target options in each DAO message they transmit. One RPL Target option MUST have a prefix that includes the node's IPv6 address if that node needs the DODAG to provision Downward routes to that node. The RPL Target option MAY be immediately followed by an opaque RPL Target Descriptor option that qualifies it.

2. When a node updates the information in a Transit Information option for a Target option that covers one of its addresses, it MUST increment the Path Sequence number in that Transit Information option. The Path Sequence number MAY be incremented occasionally to cause a refresh to the Downward routes.

3. One or more RPL Target options in a unicast DAO message MUST be followed by one or more Transit Information options. All the transit options apply to all the Target options that immediately precede them.

4. Multicast DAOs MUST NOT include the DODAG Parent Address subfield in Transit Information options.

5. A node that receives and processes a DAO message containing information for a specific Target, and that has prior information for that Target, MUST use the Path Sequence number in the Transit Information option associated with that Target in order to determine whether or not the DAO message contains updated information per Section 7.

6. If a node receives a DAO message that does not follow the above rules, it MUST discard the DAO message without further processing.

In Non-Storing mode, the root builds a strict source routing header, hop-by-hop, by recursively looking up one-hop information that ties a Target (address or prefix) and a transit address together. In some cases, when a child address is derived from a prefix that is owned and advertised by a parent, that parent-child relationship may be inferred by the root for the purpose of constructing the source routing header. In all other cases, it is necessary to inform the root of the transit-Target relationship from a reachable target, so as to later enable the recursive construction of the routing header. An address that is advertised as a Target in a DAO message MUST be collocated in the same router, or reachable on-link by the router

that owns the address that is indicated in the associated Transit Information. The following additional rules apply to ensure the continuity of the end-to-end source route path:

1. The address of a parent used in the transit option MUST be taken from a PIO from that parent with the 'R' flag set. The 'R' flag in a PIO indicates that the prefix field actually contains the full parent address but the child SHOULD NOT assume that the parent address is on-link.

2. A PIO with an 'A' flag set indicates that the RPL child node may use the prefix to autoconfigure an address. A parent that advertises a prefix in a PIO with the 'A' flag set MUST ensure that the address or the whole prefix in the PIO is reachable from the root by advertising it as a DAO target. If the parent also sets the 'L' flag indicating that the prefix is on-link, then it MUST advertise the whole prefix as Target in a DAO message. If the 'L' flag is cleared and the 'R' flag is set, indicating that the parent provides its own address in the PIO, then the parent MUST advertise that address as a DAO target.

3. An address that is advertised as Target in a DAO message MUST be collocated in the same router or reachable on-link by the router that owns the address that is indicated in the associated Transit Information.

4. In order to enable an optimum compression of the routing header, the parent SHOULD set the 'R' flag in all PIOs with the 'A' flag set and the 'L' flag cleared, and the child SHOULD prefer to use as transit the address of the parent that is found in the PIO that is used to autoconfigure the address that is advertised as Target in the DAO message.

5. A router might have targets that are not known to be on-link for a parent, either because they are addresses located on an alternate interface or because they belong to nodes that are external to RPL, for instance connected hosts. In order to inject such a Target in the RPL network, the router MUST advertise itself as the DODAG Parent Address subfield in the Transit Information option for that target, using an address that is on-link for that nodes DAO parent. If the Target belongs to an external node, then the router MUST set the External 'E' flag in the Transit Information.

A child node that has autoconfigured an address from a parent PIO with the 'L' flag set does not need to advertise that address as a DAO Target since the parent ensures that the whole prefix is already reachable from the root. However, if the 'L' flag is not set, then

it is necessary, in Non-Storing mode, for the child node to inform the root of the parent-child relationship, using a reachable address of the parent, so as to enable the recursive construction of the routing header. This is done by associating an address of the parent as transit with the address of the child as Target in a DAO message.

9.5. DAO Transmission Scheduling

Because DAOs flow Upward, receiving a unicast DAO can trigger sending a unicast DAO to a DAO parent.

1. On receiving a unicast DAO message with updated information, such as containing a Transit Information option with a new Path Sequence, a node SHOULD send a DAO. It SHOULD NOT send this DAO message immediately. It SHOULD delay sending the DAO message in order to aggregate DAO information from other nodes for which it is a DAO parent.

2. A node SHOULD delay sending a DAO message with a timer (DelayDAO). Receiving a DAO message starts the DelayDAO timer. DAO messages received while the DelayDAO timer is active do not reset the timer. When the DelayDAO timer expires, the node sends a DAO.

3. When a node adds a node to its DAO parent set, it SHOULD schedule a DAO message transmission.

DelayDAO's value and calculation is implementation dependent. A default value of DEFAULT_DAO_DELAY is defined in this specification.

9.6. Triggering DAO Messages

Nodes can trigger their sub-DODAG to send DAO messages. Each node maintains a DAO Trigger Sequence Number (DTSN), which it communicates through DIO messages.

1. If a node hears one of its DAO parents increment its DTSN, the node MUST schedule a DAO message transmission using rules in Sections 9.3 and 9.5.

2. In Non-Storing mode, if a node hears one of its DAO parents increment its DTSN, the node MUST increment its own DTSN.

In a Storing mode of operation, as part of routine routing table updates and maintenance, a storing node MAY increment DTSN in order to reliably trigger a set of DAO updates from its immediate children.

In a Storing mode of operation, it is not necessary to trigger DAO updates from the entire sub-DODAG, since that state information will propagate hop-by-hop Up the DODAG.

In a Non-Storing mode of operation, a DTSN increment will also cause the immediate children of a node to increment their DTSN in turn, triggering a set of DAO updates from the entire sub-DODAG. Typically, in a Non-Storing mode of operation, only the root would independently increment the DTSN when a DAO refresh is needed but a global repair (such as by incrementing DODAGVersionNumber) is not desired. Typically, in a Non-Storing mode of operation, all non-root nodes would increment their DTSN only when their parent(s) are observed to do so.

In general, a node may trigger DAO updates according to implementation-specific logic, such as based on the detection of a Downward route inconsistency or occasionally based upon an internal timer.

In a storing network, selecting a proper DelayDAO for triggered DAOs can greatly reduce the number of DAOs transmitted. The trigger flows Down the DODAG; in the best case, the DAOs flow Up the DODAG such that leaves send DAOs first, with each node sending a DAO message only once. Such a scheduling could be approximated by setting DelayDAO inversely proportional to Rank. Note that this suggestion is intended as an optimization to allow efficient aggregation (it is not required for correct operation in the general case).

9.7. Non-Storing Mode

In Non-Storing mode, RPL routes messages Downward using IP source routing. The following rule applies to nodes that are in Non-Storing mode. Storing mode has a separate set of rules, described in Section 9.8.

1. The DODAG Parent Address subfield of a Transit Information option MUST contain one or more addresses. All of these addresses MUST be addresses of DAO parents of the sender.

2. DAOs are sent directly to the root along a default route installed as part of the parent selection.

3. When a node removes a node from its DAO parent set, it MAY generate a new DAO message with an updated Transit Information option.

In Non-Storing mode, a node uses DAOs to report its DAO parents to the DODAG root. The DODAG root can piece together a Downward route to a node by using DAO parent sets from each node in the route. The Path Sequence information may be used to detect stale DAO information. The purpose of this per-hop route calculation is to minimize traffic when DAO parents change. If nodes reported complete source routes, then on a DAO parent change, the entire sub-DODAG would have to send new DAOs to the DODAG root. Therefore, in Non-Storing mode, a node can send a single DAO, although it might choose to send more than one DAO message to each of multiple DAO parents.

Nodes pack DAOs by sending a single DAO message with multiple RPL Target options. Each RPL Target option has its own, immediately following, Transit Information options.

9.8. Storing Mode

In Storing mode, RPL routes messages Downward by the IPv6 destination address. The following rules apply to nodes that are in Storing mode:

1. The DODAG Parent Address subfield of a Transmit Information option MUST be empty.

2. On receiving a unicast DAO, a node MUST compute if the DAO would change the set of prefixes that the node itself advertises. This computation SHOULD include consultation of the Path Sequence information in the Transit Information options associated with the DAO, to determine if the DAO message contains newer information that supersedes the information already stored at the node. If so, the node MUST generate a new DAO message and transmit it, following the rules in Section 9.5. Such a change includes receiving a No-Path DAO.

3. When a node generates a new DAO, it SHOULD unicast it to each of its DAO parents. It MUST NOT unicast the DAO message to nodes that are not DAO parents.

4. When a node removes a node from its DAO parent set, it SHOULD send a No-Path DAO message (Section 6.4.3) to that removed DAO parent to invalidate the existing route.

5. If messages to an advertised Downward address suffer from a forwarding error, Neighbor Unreachable Detection (NUD), or similar failure, a node MAY mark the address as unreachable and generate an appropriate No-Path DAO.

DAOs advertise to which destination addresses and prefixes a node has routes. Unlike in Non-Storing mode, these DAOs do not communicate information about the routes themselves: that information is stored within the network and is implicit from the IPv6 source address. When a storing node generates a DAO, it uses the stored state of DAOs it has received to produce a set of RPL Target options and their associated Transmit Information options.

Because this information is stored within each node's routing tables, in Storing mode, DAOs are communicated directly to DAO parents, who store this information.

9.9. Path Control

A DAO message from a node contains one or more Target options. Each Target option specifies either a prefix advertised by the node, a prefix of addresses reachable outside the LLN, the address of a destination in the node's sub-DODAG, or a multicast group to which a node in the sub-DODAG is listening. The Path Control field of the Transit Information option allows nodes to request or allow for multiple Downward routes. A node constructs the Path Control field of a Transit Information option as follows:

1. The bit width of the Path Control field MUST be equal to the value (PCS + 1), where PCS is specified in the control field of the DODAG Configuration option. Bits greater than or equal to the value (PCS + 1) MUST be cleared on transmission and MUST be ignored on reception. Bits below that value are considered "active" bits.

2. The node MUST logically construct groupings of its DAO parents while populating the Path Control field, where each group consists of DAO parents of equal preference. Those groups MUST then be ordered according to preference, which allows for a logical mapping of DAO parents onto Path Control subfields (see Figure 27). Groups MAY be repeated in order to extend over the entire bit width of the patch control field, but the order, including repeated groups, MUST be retained so that preference is properly communicated.

3. For a RPL Target option describing a node's own address or a prefix outside the LLN, at least one active bit of the Path Control field MUST be set. More active bits of the Path Control field MAY be set.

4. If a node receives multiple DAOs with the same RPL Target option, it MUST bitwise-OR the Path Control fields it receives. This aggregated bitwise-OR represents the number of Downward routes the prefix requests.

5. When a node sends a DAO message to one of its DAO parents, it MUST select one or more of the bits that are set active in the subfield that is mapped to the group containing that DAO parent from the aggregated Path Control field. A given bit can only be presented as active to one parent. The DAO message it transmits to its parent MUST have these active bits set and all other active bits cleared.

6. For the RPL Target option and DAOSequence number, the DAOs a node sends to different DAO parents MUST have disjoint sets of active Path Control bits. A node MUST NOT set the same active bit on DAOs to two different DAO parents.

7. Path Control bits SHOULD be allocated according to the preference mapping of DAO parents onto Path Control subfields, such that the active Path Control bits, or groupings of bits, that belong to a particular Path Control subfield are allocated to DAO parents within the group that was mapped to that subfield.

8. In a Non-Storing mode of operation, a node MAY pass DAOs through without performing any further processing on the Path Control field.

9. A node MUST NOT unicast a DAO message that has no active bits in the Path Control field set. It is possible that, for a given Target option, a node does not have enough aggregate Path Control bits to send a DAO message containing that Target to each of its DAO parents, in which case those least preferred DAO Parents may not get a DAO message for that Target.

The Path Control field allows a node to bound how many Downward routes will be generated to it. It sets a number of bits in the Path Control field equal to the maximum number of Downward routes it prefers. At most, each bit is sent to one DAO parent; clusters of bits can be sent to a single DAO parent for it to divide among its own DAO parents.

A node that provisions a DAO route for a Target that has an associated Path Control field SHOULD use the content of that Path Control field in order to determine an order of preference among multiple alternative DAO routes for that Target. The Path Control field assignment is derived from preference (of the DAO parents), as determined on the basis of this node's best knowledge of the "end-to-

end" aggregated metrics in the Downward direction as per the Objective Function. In Non-Storing mode the root can determine the Downward route by aggregating the information from each received DAO, which includes the Path Control indications of preferred DAO parents.

9.9.1. Path Control Example

Suppose that there is an LLN operating in Storing mode that contains a Node N with four parents, P1, P2, P3, and P4. Let N have three children, C1, C2, and C3 in its sub-DODAG. Let PCS be 7, such that there will be 8 active bits in the Path Control field: 11111111b. Consider the following example:

The Path Control field is split into four subfields, PC1 (11000000b), PC2 (00110000b), PC3 (00001100b), and PC4 (00000011b), such that those four subfields represent four different levels of preference per Figure 27. The implementation at Node N, in this example, groups {P1, P2} to be of equal preference to each other and the most preferred group overall. {P3} is less preferred to {P1, P2}, and more preferred to {P4}. Let Node N then perform its Path Control mapping such that:

```
{P1, P2} -> PC1 (11000000b) in the Path Control field
{P3}     -> PC2 (00110000b) in the Path Control field
{P4}     -> PC3 (00001100b) in the Path Control field
{P4}     -> PC4 (00000011b) in the Path Control field
```

Note that the implementation repeated {P4} in order to get complete coverage of the Path Control field.

1. Let C1 send a DAO containing a Target T with a Path Control 10000000b. Node N stores an entry associating 10000000b with the Path Control field for C1 and Target T.

2. Let C2 send a DAO containing a Target T with a Path Control 00010000b. Node N stores an entry associating 00010000b with the Path Control field for C1 and Target T.

3. Let C3 send a DAO containing a Target T with a Path Control 00001100b. Node N stores an entry associating 00001100b with the Path Control field for C1 and Target T.

4. At some later time, Node N generates a DAO for Target T. Node N will construct an aggregate Path Control field by ORing together the contribution from each of its children that have given a DAO for Target T. Thus, the aggregate Path Control field has the active bits set as: 10011100b.

5. Node N then distributes the aggregate Path Control bits among its parents P1, P2, P3, and P4 in order to prepare the DAO messages.

6. P1 and P2 are eligible to receive active bits from the most preferred subfield (11000000b). Those bits are 10000000b in the aggregate Path Control field. Node N must set the bit to one of the two parents only. In this case, Node P1 is allocated the bit and gets the Path Control field 10000000b for its DAO. There are no bits left to allocate to Node P2; thus, Node P2 would have a Path Control field of 00000000b and a DAO cannot be generated to Node P2 since there are no active bits.

7. The second-most preferred subfield (00110000b) has the active bits 00010000b. Node N has mapped P3 to this subfield. Node N may allocates the active bit to P3, constructing a DAO for P3 containing Target T with a Path Control of 00010000b.

8. The third-most preferred subfield (00001100b) has the active bits 00001100b. Node N has mapped P4 to this subfield. Node N may allocate both bits to P4, constructing a DAO for P4 containing Target T with a Path Control of 00001100b.

9. The least preferred subfield (00000011b) has no active bits. Had there been active bits, those bits would have been added to the Path Control field of the DAO constructed for P4.

10. The process of populating the DAO messages destined for P1, P2, P3, P4 with other targets (other than T) proceeds according to the aggregate Path Control fields collected for those targets.

9.10. Multicast Destination Advertisement Messages

A special case of DAO operation, distinct from unicast DAO operation, is multicast DAO operation that may be used to populate '1-hop' routing table entries.

1. A node MAY multicast a DAO message to the link-local scope all-RPL-nodes multicast address.

2. A multicast DAO message MUST be used only to advertise information about the node itself, i.e., prefixes directly connected to or owned by the node, such as a multicast group that the node is subscribed to or a global address owned by the node.

3. A multicast DAO message MUST NOT be used to relay connectivity information learned (e.g., through unicast DAO) from another node.

4. A node MUST NOT perform any other DAO-related processing on a received multicast DAO message; in particular, a node MUST NOT perform the actions of a DAO parent upon receipt of a multicast DAO.

- The multicast DAO may be used to enable direct P2P communication, without needing the DODAG to relay the packets.

10. Security Mechanisms

This section describes the generation and processing of secure RPL messages. The high-order bit of the RPL message code identifies whether or not a RPL message is secure. In addition to secure versions of basic control messages (DIS, DIO, DAO, DAO-ACK), RPL has several messages that are relevant only in networks that are security enabled.

Implementation complexity and size is a core concern for LLNs such that it may be economically or physically impossible to include sophisticated security provisions in a RPL implementation. Furthermore, many deployments can utilize link-layer or other security mechanisms to meet their security requirements without requiring the use of security in RPL.

Therefore, the security features described in this document are OPTIONAL to implement. A given implementation MAY support a subset (including the empty set) of the described security features, for example, it could support integrity and confidentiality, but not signatures. An implementation SHOULD clearly specify which security mechanisms are supported, and it is RECOMMENDED that implementers carefully consider security requirements and the availability of security mechanisms in their network.

10.1. Security Overview

RPL supports three security modes:

- Unsecured. In this security mode, RPL uses basic DIS, DIO, DAO, and DAO-ACK messages, which do not have Security sections. As a network could be using other security mechanisms, such as link-layer security, unsecured mode does not imply all messages are sent without any protection.

- Preinstalled. In this security mode, RPL uses secure messages. To join a RPL Instance, a node must have a preinstalled key. Nodes use this to provide message confidentiality, integrity, and authenticity. A node may, using this preinstalled key, join the RPL network as either a host or a router.

- Authenticated. In this security mode, RPL uses secure messages. To join a RPL Instance, a node must have a preinstalled key. Nodes use this key to provide message confidentiality, integrity, and authenticity. Using this preinstalled key, a node may join the network as a host only. To join the network as a router, a node must obtain a second key from a key authority. This key authority can authenticate that the requester is allowed to be a router before providing it with the second key. Authenticated mode cannot be supported by symmetric algorithms. As of the writing of this specification, RPL supports only symmetric algorithms: authenticated mode is included for the benefit of potential future cryptographic primitives. See Section 10.3.

Whether or not the RPL Instance uses unsecured mode is signaled by whether it uses secure RPL messages. Whether a secured network uses the preinstalled or authenticated mode is signaled by the 'A' bit of the DAG Configuration option.

This specification specifies CCM -- Counter with CBC-MAC (Cipher Block Chaining - Message Authentication Code) -- as the cryptographic basis for RPL security [RFC3610]. In this specification, CCM uses AES-128 as its underlying cryptographic algorithm. There are bits reserved in the Security section to specify other algorithms in the future.

All secured RPL messages have either a MAC or a signature. Optionally, secured RPL messages also have encryption protection for confidentiality. Secured RPL message formats support both integrated encryption/authentication schemes (e.g., CCM) as well as schemes that separately encrypt and authenticate packets.

10.2. Joining a Secure Network

RPL security assumes that a node wishing to join a secured network has been pre-configured with a shared key for communicating with neighbors and the RPL root. To join a secure RPL network, a node either listens for secure DIOs or triggers secure DIOs by sending a secure DIS. In addition to the DIO/DIS rules in Section 8, secure DIO and DIS messages have these rules:

1. If sent, this initial secure DIS MUST set the Key Identifier Mode field to 0 (00) and MUST set the Security Level field to 1 (001). The key used MUST be the pre-configured group key (Key Index 0x00).

2. When a node resets its Trickle timer in response to a secure DIS (Section 8.3), the next DIO it transmits MUST be a secure DIO with the same security configuration as the secure DIS. If a

node receives multiple secure DIS messages before it transmits a DIO, the secure DIO MUST have the same security configuration as the last DIS to which it is responding.

3. When a node sends a DIO in response to a unicast secure DIS (Section 8.3), the DIO MUST be a secure DIO.

The above rules allow a node to join a secured RPL Instance using the pre-configured shared key. Once a node has joined the DODAG using the pre-configured shared key, the 'A' bit of the Configuration option determines its capabilities. If the 'A' bit of the Configuration option is cleared, then nodes can use this preinstalled, shared key to exchange messages normally: it can issue DIOs, DAOs, etc.

If the 'A' bit of the Configuration option is set and the RPL Instance is operating in authenticated mode:

1. A node MUST NOT advertise a Rank besides INFINITE_RANK in secure DIOs secured with Key Index 0x00. When processing DIO messages secured with Key Index 0x00, a processing node MUST consider the advertised Rank to be INFINITE_RANK. Any other value results in the message being discarded.

2. Secure DAOs using a Key Index 0x00 MUST NOT have a RPL Target option with a prefix besides the node's address. If a node receives a secured DAO message using the preinstalled, shared key where the RPL Target option does not match the IPv6 source address, it MUST discard the secured DAO message without further processing.

The above rules mean that in RPL Instances where the 'A' bit is set, using Key Index 0x00, a node can join the RPL Instance as a host but not a router. A node must communicate with a key authority to obtain a key that will enable it to act as a router.

10.3. Installing Keys

Authenticated mode requires a would-be router to dynamically install new keys once they have joined a network as a host. Having joined as a host, the node uses standard IP messaging to communicate with an authorization server, which can provide new keys.

The protocol to obtain such keys is out of scope for this specification and to be elaborated in future specifications. That elaboration is required for RPL to securely operate in authenticated mode.

10.4. Consistency Checks

RPL nodes send Consistency Check (CC) messages to protect against replay attacks and synchronize counters.

1. If a node receives a unicast CC message with the 'R' bit cleared, and it is a member of or is in the process of joining the associated DODAG, it SHOULD respond with a unicast CC message to the sender. This response MUST have the 'R' bit set, and it MUST have the same CC nonce, RPLInstanceID, and DODAGID fields as the message it received.

2. If a node receives a multicast CC message, it MUST discard the message with no further processing.

Consistency Check messages allow nodes to issue a challenge-response to validate a node's current counter value. Because the CC nonce is generated by the challenger, an adversary replaying messages is unlikely to be able to generate a correct response. The counter in the Consistency Check response allows the challenger to validate the counter values it hears.

10.5. Counters

In the simplest case, the counter value is an unsigned integer that a node increments by one or more on each secured RPL transmission. The counter MAY represent a timestamp that has the following properties:

1. The timestamp MUST be at least six octets long.

2. The timestamp MUST be in 1024 Hz (binary millisecond) granularity.

3. The timestamp start time MUST be January 1, 1970, 12:00:00AM UTC.

4. If the counter represents a timestamp, the counter value MUST be a value computed as follows. Let T be the timestamp, S be the start time of the key in use, and E be the end time of the key in use. Both S and E are represented using the same three rules as the timestamp described above. If E > T < S, then the counter is invalid and a node MUST NOT generate a packet. Otherwise, the counter value is equal to T-S.

5. If the counter represents such a timestamp, a node MAY set the 'T' flag of the Security section of secured RPL packets.

6. If the Counter field does not present such a timestamp, then a node MUST NOT set the 'T' flag.

7. If a node does not have a local timestamp that satisfies the above requirements, it MUST ignore the 'T' flag.

If a node supports such timestamps and it receives a message with the 'T' flag set, it MAY apply the temporal check on the received message described in Section 10.7.1. If a node receives a message without the 'T' flag set, it MUST NOT apply this temporal check. A node's security policy MAY, for application reasons, include rejecting all messages without the 'T' flag set.

The 'T' flag is present because many LLNs today already maintain global time synchronization at sub-millisecond granularity for security, application, and other reasons. Allowing RPL to leverage this existing functionality when present greatly simplifies solutions to some security problems, such as delay protection.

10.6. Transmission of Outgoing Packets

Given an outgoing RPL control packet and the required security protection, this section describes how RPL generates the secured packet to transmit. It also describes the order of cryptographic operations to provide the required protection.

The requirement for security protection and the level of security to be applied to an outgoing RPL packet shall be determined by the node's security policy database. The configuration of this security policy database for outgoing packet processing is implementation specific.

Where secured RPL messages are to be transmitted, a RPL node MUST set the Security section (T, Sec, KIM, and LVL) in the outgoing RPL packet to describe the protection level and security settings that are applied (see Section 6.1). The Security subfield bit of the RPL Message Code field MUST be set to indicate the secure RPL message.

The counter value used in constructing the AES-128 CCM nonce (Figure 31) to secure the outgoing packet MUST be an increment of the last counter transmitted to the particular destination address.

Where security policy specifies the application of delay protection, the Timestamp counter used in constructing the CCM nonce to secure the outgoing packet MUST be incremented according to the rules in Section 10.5. Where a Timestamp counter is applied (indicated with the 'T' flag set), the locally maintained Timestamp counter MUST be included as part of the transmitted secured RPL message.

The cryptographic algorithm used in securing the outgoing packet shall be specified by the node's security policy database and MUST be indicated in the value of the Sec field set within the outgoing message.

The security policy for the outgoing packet shall determine the applicable KIM and Key Identifier specifying the security key to be used for the cryptographic packet processing, including the optional use of signature keys (see Section 6.1). The security policy will also specify the algorithm (Algorithm) and level of protection (Level) in the form of authentication or authentication and encryption, and potential use of signatures that shall apply to the outgoing packet.

Where encryption is applied, a node MUST replace the original packet payload with that payload encrypted using the security protection, key, and CCM nonce specified in the Security section of the packet.

All secured RPL messages include integrity protection. In conjunction with the security algorithm processing, a node derives either a MAC or signature that MUST be included as part of the outgoing secured RPL packet.

10.7. Reception of Incoming Packets

This section describes the reception and processing of a secured RPL packet. Given an incoming secured RPL packet, where the Security subfield bit of the RPL Message Code field is set, this section describes how RPL generates an unencrypted variant of the packet and validates its integrity.

The receiver uses the RPL security control fields to determine the necessary packet security processing. If the described level of security for the message type and originator is unknown or does not meet locally maintained security policies, a node MUST discard the packet without further processing, MAY raise a management alert, and MUST NOT send any messages in response. These policies can include security levels, keys used, source identifiers, or the lack of timestamp-based counters (as indicated by the 'T' flag). The configuration of the security policy database for incoming packet processing is out of scope for this specification (it may, for example, be defined through DIO Configuration or through out-of-band administrative router configuration).

Where the message Security Level (LVL) indicates an encrypted RPL message, the node uses the key information identified through the KIM field as well as the CCM nonce as input to the message payload decryption processing. The CCM nonce shall be derived from the

message Counter field and other received and locally maintained information (see Section 10.9.1). The plaintext message contents shall be obtained by invoking the inverse cryptographic mode of operation specified by the Sec field of the received packet.

The receiver shall use the CCM nonce and identified key information to check the integrity of the incoming packet. If the integrity check fails against the received MAC, a node MUST discard the packet.

If the received message has an initialized (zero value) counter value and the receiver has an incoming counter currently maintained for the originator of the message, the receiver MUST initiate a counter resynchronization by sending a Consistency Check response message (see Section 6.6) to the message source. The Consistency Check response message shall be protected with the current full outgoing counter maintained for the particular node address. That outgoing counter will be included within the security section of the message while the incoming counter will be included within the Consistency Check message payload.

Based on the specified security policy, a node MAY apply replay protection for a received RPL message. The replay check SHOULD be performed before the authentication of the received packet. The counter, as obtained from the incoming packet, shall be compared against the watermark of the incoming counter maintained for the given origination node address. If the received message counter value is non-zero and less than the maintained incoming counter watermark, a potential packet replay is indicated and the node MUST discard the incoming packet.

If delay protection is specified as part of the incoming packet security policy checks, the Timestamp counter is used to validate the timeliness of the received RPL message. If the incoming message Timestamp counter value indicates a message transmission time prior to the locally maintained transmission time counter for the originator address, a replay violation is indicated and the node MUST discard the incoming packet. If the received Timestamp counter value indicates a message transmission time that is earlier than the Current time less the acceptable packet delay, a delay violation is indicated and the node MUST discard the incoming packet.

Once a message has been decrypted, where applicable, and has successfully passed its integrity check, replay check, and optionally delay-protection checks, the node can update its local security information, such as the source's expected counter value for replay comparison.

A node MUST NOT update its security information on receipt of a message that fails security policy checks or other applied integrity, replay, or delay checks.

10.7.1. Timestamp Key Checks

If the 'T' flag of a message is set and a node has a local timestamp that follows the requirements in Section 10.5, then a node MAY check the temporal consistency of the message. The node computes the transmit time of the message by adding the counter value to the start time of the associated key. If this transmit time is past the end time of the key, the node MAY discard the message without further processing. If the transmit time is too far in the past or future compared to the local time on the receiver, it MAY discard the message without further processing.

10.8. Coverage of Integrity and Confidentiality

For a RPL ICMPv6 message, the entire packet is within the scope of RPL security.

MACs and signatures are calculated over the entire unsecured IPv6 packet. When computing MACs and signatures, mutable IPv6 fields are considered to be filled with zeroes, following the rules in Section 3.3.3.1 of [RFC4302] (IPsec Authenticated Header). MAC and signature calculations are performed before any compression that lower layers may apply.

When a RPL ICMPv6 message is encrypted, encryption starts at the first byte after the Security section and continues to the last byte of the packet. The IPv6 header, ICMPv6 header, and RPL message up to the end of the Security section are not encrypted, as they are needed to correctly decrypt the packet.

For example, a node sending a message with LVL=1, KIM=0, and Algorithm=0 uses the CCM algorithm [RFC3610] to create a packet with attributes ENC-MAC-32: it encrypts the packet and appends a 32-bit MAC. The block cipher key is determined by the Key Index. The CCM nonce is computed as described in Section 10.9.1; the message to authenticate and encrypt is the RPL message starting at the first byte after the Security section and ends with the last byte of the packet. The additional authentication data starts with the beginning of the IPv6 header and ends with the last byte of the RPL Security section.

10.9. Cryptographic Mode of Operation

The cryptographic mode of operation described in this specification (Algorithm = 0) is based on CCM and the block-cipher AES-128 [RFC3610]. This mode of operation is widely supported by existing implementations. CCM mode requires a nonce (CCM nonce).

10.9.1. CCM Nonce

A RPL node constructs a CCM nonce as follows:

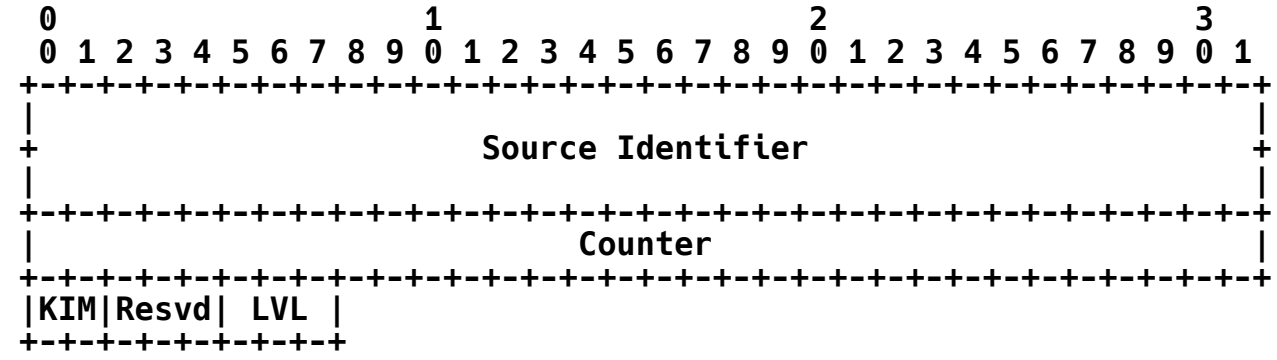


Figure 31: CCM Nonce

Source Identifier: 8 bytes. Source Identifier is set to the logical identifier of the originator of the protected packet.

Counter: 4 bytes. Counter is set to the (uncompressed) value of the corresponding field in the Security option of the RPL control message.

Key Identifier Mode (KIM): 2 bits. KIM is set to the value of the corresponding field in the Security option of the RPL control message.

Security Level (LVL): 3 bits. Security Level is set to the value of the corresponding field in the Security option of the RPL control message.

Unassigned bits of the CCM nonce are reserved. They MUST be set to zero when constructing the CCM nonce.

All fields of the CCM nonce are represented in most significant octet and most significant bit first order.

10.9.2. Signatures

If the KIM indicates the use of signatures (a value of 3), then a node appends a signature to the data payload of the packet. The Security Level (LVL) field describes the length of this signature. The signature scheme in RPL for Security Mode 3 is an instantiation of the RSA algorithm (RSASSA-PSS) as defined in Section 8.1 of [RFC3447]. As public key, it uses the pair (n,e), where n is a 2048-bit or 3072-bit RSA modulus and where e=2^{16}+1. It uses CCM mode [RFC3610] as the encryption scheme with M=0 (as a stream-cipher). Note that although [RFC3610] disallows the CCM mode with M=0, RPL explicitly allows the CCM mode with M=0 when used in conjunction with a signature, because the signature provides sufficient data authentication. Here, the CCM mode with M=0 is specified as in [RFC3610], but where the M' field in Section 2.2 MUST be set to 0. It uses the SHA-256 hash function specified in Section 6.2 of [FIPS180]. It uses the message encoding rules of Section 8.1 of [RFC3447].

Let 'a' be a concatenation of a 6-byte representation of counter and the message header. The packet payload is the right-concatenation of packet data 'm' and the signature 's'. This signature scheme is invoked with the right-concatenation of the message parts a and m, whereas the signature verification is invoked with the right-concatenation of the message parts a and m and with signature s.

RSA signatures of this form provide sufficient protection for RPL networks. If needed, alternative signature schemes that produce more concise signatures is out of scope for this specification and may be the subject of a future specification.

An implementation that supports RSA signing with either 2048-bit or 3072-bit signatures SHOULD support verification of both 2048-bit and 3072-bit RSA signatures. This is in consideration of providing an upgrade path for a RPL deployment.

11. Packet Forwarding and Loop Avoidance/Detection

11.1. Suggestions for Packet Forwarding

This document specifies a routing protocol. These non-normative suggestions are provided to aid in the design of a forwarding implementation by illustrating how such an implementation could work with RPL.

When forwarding a packet to a destination, precedence is given to selection of a next-hop successor as follows:

1. This specification only covers how a successor is selected from the DODAG Version that matches the RPLInstanceID marked in the IPv6 header of the packet being forwarded. Routing outside the instance can be done as long as additional rules are put in place such as strict ordering of instances and routing protocols to protect against loops. Such rules may be defined in a separate document.

2. If a local administrative preference favors a route that has been learned from a different routing protocol than RPL, then use that successor.

3. If the packet header specifies a source route by including an RH4 header as specified in [RFC6554], then use that route. If the node fails to forward the packet with that specified source route, then that packet should be dropped. The node MAY log an error. The node may send an ICMPv6 error in Source Routing Header message to the source of the packet (see Section 20.18).

4. If there is an entry in the routing table matching the destination that has been learned from a multicast destination advertisement (e.g., the destination is a one-hop neighbor), then use that successor.

5. If there is an entry in the routing table matching the destination that has been learned from a unicast destination advertisement (e.g., the destination is located Down the sub-DODAG), then use that successor. If there are DAO Path Control bits associated with multiple successors, then consult the Path Control bits to order the successors by preference when choosing. If, for a given DAO Path Control bit, multiple successors are recorded as having asserted that bit, precedence should be given to the successor who most recently asserted that bit.

6. If there is a DODAG Version offering a route to a prefix matching the destination, then select one of those DODAG parents as a successor according to the OF and routing metrics.

7. Any other as-yet-unattempted DODAG parent may be chosen for the next attempt to forward a unicast packet when no better match exists.

8. Finally, the packet is dropped. ICMP Destination Unreachable MAY be invoked (an inconsistency is detected).

Hop Limit MUST be decremented when forwarding per [RFC2460].

Note that the chosen successor MUST NOT be the neighbor that was the predecessor of the packet (split horizon), except in the case where it is intended for the packet to change from an Upward to a Downward direction, as determined by the routing table of the node making the change, such as switching from DIO routes to DAO routes as the destination is neared in order to continue traveling toward the destination.

11.2. Loop Avoidance and Detection

RPL loop avoidance mechanisms are kept simple and designed to minimize churn and states. Loops may form for a number of reasons, e.g., control packet loss. RPL includes a reactive loop detection technique that protects from meltdown and triggers repair of broken paths.

RPL loop detection uses RPL Packet Information that is transported within the data packets, relying on an external mechanism such as [RFC6553] that places in the RPL Packet Information in an IPv6 Hop-by-Hop option header.

The content of RPL Packet Information is defined as follows:

Down 'O': 1-bit flag indicating whether the packet is expected to progress Up or Down. A router sets the 'O' flag when the packet is expected to progress Down (using DAO routes), and clears it when forwarding toward the DODAG root (to a node with a lower Rank). A host or RPL leaf node MUST set the 'O' flag to 0.

Rank-Error 'R': 1-bit flag indicating whether a Rank error was detected. A Rank error is detected when there is a mismatch in the relative Ranks and the direction as indicated in the 'O' bit. A host or RPL leaf node MUST set the 'R' bit to 0.

Forwarding-Error 'F': 1-bit flag indicating that this node cannot forward the packet further towards the destination. The 'F' bit might be set by a child node that does not have a route to destination for a packet with the Down 'O' bit set. A host or RPL leaf node MUST set the 'F' bit to 0.

RPLInstanceID: 8-bit field indicating the DODAG instance along which the packet is sent.

SenderRank: 16-bit field set to zero by the source and to DAGRank(rank) by a router that forwards inside the RPL network.

11.2.1. Source Node Operation

If the source is aware of the RPLInstanceID that is preferred for the packet, then it MUST set the RPLInstanceID field associated with the packet accordingly; otherwise, it MUST set it to the RPL_DEFAULT_INSTANCE.

11.2.2. Router Operation

11.2.2.1. Instance Forwarding

The RPLInstanceID is associated by the source with the packet. This RPLInstanceID MUST match the RPL Instance onto which the packet is placed by any node, be it a host or router. The RPLInstanceID is part of the RPL Packet Information.

A RPL router that forwards a packet in the RPL network MUST check if the packet includes the RPL Packet Information. If not, then the RPL router MUST insert the RPL Packet Information. If the router is an ingress router that injects the packet into the RPL network, the router MUST set the RPLInstanceID field in the RPL Packet Information. The details of how that router determines the mapping to a RPLInstanceID are out of scope for this specification and left to future specification.

A router that forwards a packet outside the RPL network MUST remove the RPL Packet Information.

When a router receives a packet that specifies a given RPLInstanceID and the node can forward the packet along the DODAG associated to that instance, then the router MUST do so and leave the RPLInstanceID value unchanged.

If any node cannot forward a packet along the DODAG associated with the RPLInstanceID, then the node SHOULD discard the packet and send an ICMP error message.

11.2.2.2. DAG Inconsistency Loop Detection

The DODAG is inconsistent if the direction of a packet does not match the Rank relationship. A receiver detects an inconsistency if it receives a packet with either:

- the 'O' bit set (to Down) from a node of a higher Rank.
- the 'O' bit cleared (for Up) from a node of a lower Rank.

When the DODAG root increments the DODAGVersionNumber, a temporary Rank discontinuity may form between the next DODAG Version and the prior DODAG Version, in particular, if nodes are adjusting their Rank in the next DODAG Version and deferring their migration into the next DODAG Version. A router that is still a member of the prior DODAG Version may choose to forward a packet to a (future) parent that is in the next DODAG Version. In some cases, this could cause the parent to detect an inconsistency because the Rank-ordering in the prior DODAG Version is not necessarily the same as in the next DODAG Version, and the packet may be judged not to be making forward progress. If the sending router is aware that the chosen successor has already joined the next DODAG Version, then the sending router MUST update the SenderRank to INFINITE_RANK as it forwards the packets across the discontinuity into the next DODAG Version in order to avoid a false detection of Rank inconsistency.

One inconsistency along the path is not considered a critical error and the packet may continue. However, a second detection along the path of the same packet should not occur and the packet MUST be dropped.

This process is controlled by the Rank-Error bit associated with the packet. When an inconsistency is detected on a packet, if the Rank-Error bit was not set, then the Rank-Error bit is set. If it was set the packet MUST be discarded and the Trickle timer MUST be reset.

11.2.2.3. DAO Inconsistency Detection and Recovery

DAO inconsistency loop recovery is a mechanism that applies to Storing mode of operation only.

In Non-Storing mode, the packets are source routed to the destination, and DAO inconsistencies are not corrected locally. Instead, an ICMP error with a new code "Error in Source Routing Header" is sent back to the root. The "Error in Source Routing Header" message has the same format as the "Destination Unreachable Message", as specified in [RFC4443]. The portion of the invoking packet that is sent back in the ICMP message should record at least up to the routing header, and the routing header should be consumed by this node so that the destination in the IPv6 header is the next hop that this node could not reach.

A DAO inconsistency happens when a router has a Downward route that was previously learned from a DAO message via a child, but that Downward route is not longer valid in the child, e.g., because that related state in the child has been cleaned up. With DAO inconsistency loop recovery, a packet can be used to recursively explore and clean up the obsolete DAO states along a sub-DODAG.

In a general manner, a packet that goes Down should never go Up again. If DAO inconsistency loop recovery is applied, then the router SHOULD send the packet back to the parent that passed it with the Forwarding-Error 'F' bit set and the 'O' bit left untouched. Otherwise, the router MUST silently discard the packet.

Upon receiving a packet with a Forwarding-Error bit set, the node MUST remove the routing states that caused forwarding to that neighbor, clear the Forwarding-Error bit, and attempt to send the packet again. The packet may be sent to an alternate neighbor, after the expiration of a user-configurable implementation-specific timer. If that alternate neighbor still has an inconsistent DAO state via this node, the process will recurse, this node will set the Forwarding-Error 'F' bit, and the routing state in the alternate neighbor will be cleaned up as well.

12. Multicast Operation

This section describes a multicast routing operation over an IPv6 RPL network and, specifically, how unicast DAOs can be used to relay group registrations. The same DODAG construct can be used to forward unicast and multicast traffic. This section is limited to a description of how group registrations may be exchanged and how the forwarding infrastructure operates. It does not provide a full description of multicast within an LLN and, in particular, does not describe the generation of DODAGs specifically targeted at multicast or the details of operating RPL for multicast -- that will be the subject of further specifications.

The multicast group registration uses DAO messages that are identical to unicast except for the type of address that is transported. The main difference is that the multicast traffic going down is copied to all the children that have registered with the multicast group, whereas unicast traffic is passed to one child only.

Nodes that support the RPL Storing mode of operation SHOULD also support multicast DAO operations as described below. Nodes that only support the Non-Storing mode of operation are not expected to support this section.

The multicast operation is controlled by the MOP field in the DIO.

- If the MOP field requires multicast support, then a node that joins the RPL network as a router must operate as described in this section for multicast signaling and forwarding within the RPL network. A node that does not support the multicast operation required by the MOP field can only join as a leaf.

- If the MOP field does not require multicast support, then multicast is handled by some other way that is out of scope for this specification. (Examples may include a series of unicast copies or limited-scope flooding).

A router might select to pass a listener registration DAO message to its preferred parent only; in which case, multicast packets coming back might be lost for all of its sub-DODAGs if the transmission fails over that link. Alternatively, the router might select copying additional parents as it would do for DAO messages advertising unicast destinations; in which case, there might be duplicates that the router will need to prune.

As a result, multicast routing states are installed in each router on the way from the listeners to the DODAG root, enabling the root to copy a multicast packet to all its children routers that had issued a DAO message including a Target option for that multicast group.

For a multicast packet sourced from inside the DODAG, the packet is passed to the preferred parents, and if that fails, then to the alternates in the DODAG. The packet is also copied to all the registered children, except for the one that passed the packet. Finally, if there is a listener in the external infrastructure, then the DODAG root has to further propagate the packet into the external infrastructure.

As a result, the DODAG root acts as an automatic proxy Rendezvous Point for the RPL network and as source towards the non-RPL domain for all multicast flows started in the RPL domain. So, regardless of whether the root is actually attached to a non-RPL domain, and regardless of whether the DODAG is grounded or floating, the root can serve inner multicast streams at all times.

13. Maintenance of Routing Adjacency

The selection of successors, along the default paths Up along the DODAG, or along the paths learned from destination advertisements Down along the DODAG, leads to the formation of routing adjacencies that require maintenance.

In IGPs, such as OSPF [RFC4915] or IS-IS [RFC5120], the maintenance of a routing adjacency involves the use of keepalive mechanisms (Hellos) or other protocols such as the Bidirectional Forwarding Detection (BFD) [RFC5881] and the MANET Neighborhood Discovery Protocol (NHDP) [RFC6130]. Unfortunately, such a proactive approach is often not desirable in constrained environments where it would lead to excessive control traffic in light of the data traffic with a negative impact on both link loads and nodes resources.

By contrast with those routing protocols, RPL does not define any keepalive mechanisms to detect routing adjacency failures: this is because in many cases, such a mechanism would be too expensive in terms of bandwidth and, even more importantly, energy (a battery-operated device could not afford to send periodic keepalives). Still RPL requires an external mechanisms to detect that a neighbor is no longer reachable. Such a mechanism should preferably be reactive to traffic in order to minimize the overhead to maintain the routing adjacency and focus on links that are actually being used.

Example reactive mechanisms that can be used include:

The Neighbor Unreachability Detection [RFC4861] mechanism.

Layer 2 triggers [RFC5184] derived from events such as association states and L2 acknowledgements.

14. Guidelines for Objective Functions

An Objective Function (OF), in conjunction with routing metrics and constraints, allows for the selection of a DODAG to join, and a number of peers in that DODAG as parents. The OF is used to compute an ordered list of parents. The OF is also responsible to compute the Rank of the device within the DODAG Version.

The Objective Function is indicated in the DIO message using an Objective Code Point (OCP), and it indicates the method that must be used to construct the DODAG. The Objective Code Points are specified in [RFC6552] and related companion specifications.

14.1. Objective Function Behavior

Most Objective Functions are expected to follow the same abstract behavior at a node:

- The parent selection is triggered each time an event indicates that a potential next-hop information is updated. This might happen upon the reception of a DIO message, a timer elapse, all DODAG parents are unavailable, or a trigger indicating that the state of a candidate neighbor has changed.

- An OF scans all the interfaces on the node. Although, there may typically be only one interface in most application scenarios, there might be multiple of them and an interface might be configured to be usable or not for RPL operation. An interface can also be configured with a preference or dynamically learned to be better than another by some heuristics that might be link-layer dependent and are out of scope for this specification. Finally,

an interface might or might not match a required criterion for an Objective Function, for instance, a degree of security. As a result, some interfaces might be completely excluded from the computation, for example, if those interfaces cannot satisfy some advertised constraints, while others might be more or less preferred.

- An OF scans all the candidate neighbors on the possible interfaces to check whether they can act as a router for a DODAG. There might be many of them and a candidate neighbor might need to pass some validation tests before it can be used. In particular, some link layers require experience on the activity with a router to enable the router as a next hop.

- An OF computes Rank of a node for comparison by adding to the Rank of the candidate a value representing the relative locations of the node and the candidate in the DODAG Version.

  * The increase in Rank must be at least MinHopRankIncrease.

  * To keep loop avoidance and metric optimization in alignment, the increase in Rank should reflect any increase in the metric value. For example, with a purely additive metric, such as ETX, the increase in Rank can be made proportional to the increase in the metric.

  * Candidate neighbors that would cause the Rank of the node to increase are not considered for parent selection.

- Candidate neighbors that advertise an OF incompatible with the set of OFs specified by the policy functions are ignored.

- As it scans all the candidate neighbors, the OF keeps the current best parent and compares its capabilities with the current candidate neighbor. The OF defines a number of tests that are critical to reach the objective. A test between the routers determines an order relation.

  * If the routers are equal for that relation, then the next test is attempted between the routers,

  * Else the best of the two routers becomes the current best parent, and the scan continues with the next candidate neighbor.

  * Some OFs may include a test to compare the Ranks that would result if the node joined either router.

- When the scan is complete, the preferred parent is elected and the node's Rank is computed as the preferred parent Rank plus the step in Rank with that parent.

- Other rounds of scans might be necessary to elect alternate parents. In the next rounds:

  * Candidate neighbors that are not in the same DODAG are ignored.

  * Candidate neighbors that are of greater Rank than the node are ignored.

  * Candidate neighbors of an equal Rank to the node are ignored for parent selection.

  * Candidate neighbors of a lesser Rank than the node are preferred.

15. Suggestions for Interoperation with Neighbor Discovery

This specification directly borrows the Prefix Information Option (PIO) and the Route Information Option (RIO) from IPv6 ND. It is envisioned that, as future specifications build on this base, there may be additional cause to leverage parts of IPv6 ND. This section provides some suggestions for future specifications.

First and foremost, RPL is a routing protocol. One should take great care to preserve architecture when mapping functionalities between RPL and ND. RPL is for routing only. That said, there may be persuading technical reasons to allow for sharing options between RPL and IPv6 ND in a particular implementation/deployment.

In general, the following guidelines apply:

- RPL Type codes must be allocated from the RPL Control Message Options registry.

- RPL Length fields must be expressed in units of single octets, as opposed to ND Length fields, which are expressed in units of 8 octets.

- RPL options are generally not required to be aligned to 8-octet boundaries.

- When mapping/transposing an IPv6 ND option for redistribution as a RPL option, any padding octets should be removed when possible. For example, the Prefix Length field in the PIO is sufficient to describe the length of the Prefix field. When mapping/transposing

a RPL option for redistribution as an IPv6 ND option, any such padding octets should be restored. This procedure must be unambiguous.

16. Summary of Requirements for Interoperable Implementations

This section summarizes basic interoperability and references normative text for RPL implementations operating in one of three major modes. Implementations are expected to support either no Downward routes, Non-Storing mode only, or Storing mode only. A fourth mode, operation as a leaf, is also possible.

Implementations conforming to this specification may contain different subsets of capabilities as appropriate to the application scenario. It is important for the implementer to support a level of interoperability consistent with that required by the application scenario. To this end, further guidance may be provided beyond this specification (e.g., as applicability statements), and it is understood that in some cases such further guidance may override portions of this specification.

16.1. Common Requirements

In a general case, the greatest level of interoperability may be achieved when all of the nodes in a RPL LLN are cooperating to use the same MOP, OF, metrics, and constraints, and are thus able to act as RPL routers. When a node is not capable of being a RPL router, it may be possible to interoperate in a more limited manner as a RPL leaf.

All RPL implementations need to support the use of RPL Packet Information transported within data packets (Section 11.2). One such mechanism is described in [RFC6553].

RPL implementations will need to support the use of Neighbor Unreachability Detection (NUD), or an equivalent mechanism, to maintain the reachability of neighboring RPL nodes (Section 8.2.1). Alternate mechanisms may be optimized to the constrained capabilities of the implementation, such as hints from the link layer.

This specification provides means to obtain a PIO and thus form an IPv6 address. When that mechanism is used, it may be necessary to perform address resolution and duplicate address detection through an external process, such as IPv6 ND [RFC4861] or 6LoWPAN ND [6LOWPAN-ND].

16.2. Operation as a RPL Leaf Node (Only)

- An implementation of a leaf node (only) does not ever participate as a RPL router. Interoperable implementations of leaf nodes behave as summarized in Section 8.5.
- Support of a particular MOP encoding is not required, although if the leaf node sends DAO messages to set up Downward routes, the leaf node should do so in a manner consistent with the mode of operation indicated by the MOP.
- Support of a particular OF is not required.
- In summary, a leaf node does not generally issue DIO messages, it may issue DAO and DIS messages. A leaf node accepts DIO messages though it generally ignores DAO and DIS messages.

16.3. Operation as a RPL Router

If further guidance is not available then a RPL router implementation MUST at least support the metric-less OF0 [RFC6552].

For consistent operation a RPL router implementation needs to support the MOP in use by the DODAG.

All RPL routers will need to implement Trickle [RFC6206].

16.3.1. Support for Upward Routes (Only)

An implementation of a RPL router that supports only Upward routes supports the following:

- Upward routes (Section 8)
- MOP encoding 0 (Section 20.3)
- In summary, DIO and DIS messages are issued, and DAO messages are not issued. DIO and DIS messages are accepted, and DAO messages are ignored.

16.3.2. Support for Upward Routes and Downward Routes in Non-Storing Mode

An implementation of a RPL router that supports Upward routes and Downward routes in Non-Storing mode supports the following:

- Upward routes (Section 8)

- Downward routes (Non-Storing) (Section 9)
- MOP encoding 1 (Section 20.3)
- Source-routed Downward traffic ([RFC6554])
- In summary, DIO and DIS messages are issued, and DAO messages are issued to the DODAG root. DIO and DIS messages are accepted, and DAO messages are ignored by nodes other than DODAG roots. Multicast is not supported through the means described in this specification, though it may be supported through some alternate means.

16.3.3. Support for Upward Routes and Downward Routes in Storing Mode

An implementation of a RPL router that supports Upward routes and Downward routes in Storing mode supports the following:

- Upward routes (Section 8)
- Downward routes (Storing) (Section 9)
- MOP encoding 2 (Section 20.3)
- In summary, DIO, DIS, and DAO messages are issued. DIO, DIS, and DAO messages are accepted. Multicast is not supported through the means described in this specification, though it may be supported through some alternate means.

16.3.3.1. Optional Support for Basic Multicast Scheme

A Storing mode implementation may be enhanced with basic multicast support through the following additions:

- Basic Multicast Support (Section 12)
- MOP encoding 3 (Section 20.3)

16.4. Items for Future Specification

A number of items are left to future specification, including but not limited to the following:

- How to attach a non-RPL node such as an IPv6 host, e.g., to consistently distribute at least PIO material to the attached node.

- How to obtain authentication material in support if authenticated mode is used (Section 10.3).
- Details of operation over multiple simultaneous instances.
- Advanced configuration mechanisms, such as the provisioning of RPLInstanceIDs, parameterization of Objective Functions, and parameters to control security. (It is expected that such mechanisms might extend the DIO as a means to disseminate information across the DODAG).

17. RPL Constants and Variables

The following is a summary of RPL constants and variables:

BASE_RANK: This is the Rank for a virtual root that might be used to coordinate multiple roots. BASE_RANK has a value of 0.

ROOT_RANK: This is the Rank for a DODAG root. ROOT_RANK has a value of MinHopRankIncrease (as advertised by the DODAG root), such that DAGRank(ROOT_RANK) is 1.

INFINITE_RANK: This is the constant maximum for the Rank. INFINITE_RANK has a value of 0xFFFF.

RPL_DEFAULT_INSTANCE: This is the RPLInstanceID that is used by this protocol by a node without any overriding policy. RPL_DEFAULT_INSTANCE has a value of 0.

DEFAULT_PATH_CONTROL_SIZE: This is the default value used to configure PCS in the DODAG Configuration option, which dictates the number of significant bits in the Path Control field of the Transit Information option. DEFAULT_PATH_CONTROL_SIZE has a value of 0. This configures the simplest case limiting the fan-out to 1 and limiting a node to send a DAO message to only one parent.

DEFAULT_DIO_INTERVAL_MIN: This is the default value used to configure Imin for the DIO Trickle timer. DEFAULT_DIO_INTERVAL_MIN has a value of 3. This configuration results in Imin of 8 ms.

DEFAULT_DIO_INTERVAL_DOUBLINGS: This is the default value used to configure Imax for the DIO Trickle timer. DEFAULT_DIO_INTERVAL_DOUBLINGS has a value of 20. This configuration results in a maximum interval of 2.3 hours.

DEFAULT_DIO_REDUNDANCY_CONSTANT: This is the default value used to configure k for the DIO Trickle timer. DEFAULT_DIO_REDUNDANCY_CONSTANT has a value of 10. This configuration is a conservative value for Trickle suppression mechanism.

DEFAULT_MIN_HOP_RANK_INCREASE: This is the default value of MinHopRankIncrease. DEFAULT_MIN_HOP_RANK_INCREASE has a value of 256. This configuration results in an 8-bit wide integer part of Rank.

DEFAULT_DAO_DELAY: This is the default value for the DelayDAO Timer. DEFAULT_DAO_DELAY has a value of 1 second. See Section 9.5.

DIO Timer: One instance per DODAG of which a node is a member. Expiry triggers DIO message transmission. A Trickle timer with variable interval in [0, DIOIntervalMin..2^DIOIntervalDoublings]. See Section 8.3.1

DAG Version Increment Timer: Up to one instance per DODAG of which the node is acting as DODAG root. May not be supported in all implementations. Expiry triggers increment of DODAGVersionNumber, causing a new series of updated DIO message to be sent. Interval should be chosen appropriate to propagation time of DODAG and as appropriate to application requirements (e.g., response time versus overhead).

DelayDAO Timer: Up to one timer per DAO parent (the subset of DODAG parents chosen to receive destination advertisements) per DODAG. Expiry triggers sending of DAO message to the DAO parent. See Section 9.5

RemoveTimer: Up to one timer per DAO entry per neighbor (i.e., those neighbors that have given DAO messages to this node as a DODAG parent). Expiry may trigger No-Path advertisements or immediately deallocate the DAO entry if there are no DAO parents.

18. Manageability Considerations

The aim of this section is to give consideration to the manageability of RPL, and how RPL will be operated in an LLN. The scope of this section is to consider the following aspects of manageability: configuration, monitoring, fault management, accounting, and performance of the protocol in light of the recommendations set forth in [RFC5706].

18.1. Introduction

Most of the existing IETF management standards are MIB modules (data models based on the Structure of Management Information (SMI)) to monitor and manage networking devices.

For a number of protocols, the IETF community has used the IETF Standard Management Framework, including the Simple Network Management Protocol [RFC3410], the Structure of Management Information [RFC2578], and MIB data models for managing new protocols.

As pointed out in [RFC5706], the common policy in terms of operation and management has been expanded to a policy that is more open to a set of tools and management protocols rather than strictly relying on a single protocol such as SNMP.

In 2003, the Internet Architecture Board (IAB) held a workshop on Network Management [RFC3535] that discussed the strengths and weaknesses of some IETF network management protocols and compared them to operational needs, especially configuration.

One issue discussed was the user-unfriendliness of the binary format of SNMP [RFC3410]. In the case of LLNs, it must be noted that at the time of writing, the CoRE working group is actively working on resource management of devices in LLNs. Still, it is felt that this section provides important guidance on how RPL should be deployed, operated, and managed.

As stated in [RFC5706]:

> A management information model should include a discussion of what is manageable, which aspects of the protocol need to be configured, what types of operations are allowed, what protocol-specific events might occur, which events can be counted, and for which events an operator should be notified.

These aspects are discussed in detail in the following sections.

RPL will be used on a variety of devices that may have resources such as memory varying from a few kilobytes to several hundreds of kilobytes and even megabytes. When memory is highly constrained, it may not be possible to satisfy all the requirements listed in this section. Still it is worth listing all of these in an exhaustive fashion, and implementers will then determine which of these requirements could be satisfied according to the available resources on the device.

18.2. Configuration Management

This section discusses the configuration management, listing the protocol parameters for which configuration management is relevant.

Some of the RPL parameters are optional. The requirements for configuration are only applicable for the options that are used.

18.2.1. Initialization Mode

"Architectural Principles of the Internet" [RFC1958], Section 3.8, states: "Avoid options and parameters whenever possible. Any options and parameters should be configured or negotiated dynamically rather than manually". This is especially true in LLNs where the number of devices may be large and manual configuration is infeasible. This has been taken into account in the design of RPL whereby the DODAG root provides a number of parameters to the devices joining the DODAG, thus avoiding cumbersome configuration on the routers and potential sources of misconfiguration (e.g., values of Trickle timers, etc.). Still, there are additional RPL parameters that a RPL implementation should allow to be configured, which are discussed in this section.

18.2.1.1. DIS Mode of Operation upon Boot-Up

When a node is first powered up:

1. The node may decide to stay silent, waiting to receive DIO messages from DODAG of interest (advertising a supported OF and metrics/constraints) and not send any multicast DIO messages until it has joined a DODAG.

2. The node may decide to send one or more DIS messages (optionally, requesting DIO for a specific DODAG) as an initial probe for nearby DODAGs, and in the absence of DIO messages in reply after some configurable period of time, the node may decide to root a floating DODAG and start sending multicast DIO messages.

A RPL implementation SHOULD allow configuring the preferred mode of operation listed above along with the required parameters (in the second mode: the number of DIS messages and related timer).

18.2.2. DIO and DAO Base Message and Options Configuration

RPL specifies a number of protocol parameters considering the large spectrum of applications where it will be used. That said, particular attention has been given to limiting the number of these parameters that must be configured on each RPL router. Instead, a

number of the default values can be used, and when required these parameters can be provided by the DODAG root thus allowing for dynamic parameter setting.

A RPL implementation SHOULD allow configuring the following routing protocol parameters. As pointed out above, note that a large set of parameters is configured on the DODAG root.

18.2.3. Protocol Parameters to Be Configured on Every Router in the LLN

A RPL implementation MUST allow configuring the following RPL parameters:

- RPLInstanceID [DIO message, in DIO Base message]. Although the RPLInstanceID must be configured on the DODAG root, it must also be configured as a policy on every node in order to determine whether or not the node should join a particular DODAG. Note that a second RPLInstanceID can be configured on the node, should it become root of a floating DODAG.

- List of supported Objective Code Points (OCPs)

- List of supported metrics: [RFC6551] specifies a number of metrics and constraints used for the DODAG formation. Thus, a RPL implementation should allow configuring the list of metrics that a node can accept and understand. If a DIO is received with a metric and/or constraint that is not understood or supported, as specified in Section 8.5, the node would join as a leaf node.

- Prefix Information, along with valid and preferred lifetime and the 'L' and 'A' flags. [DIO message, Prefix Information Option]. A RPL implementation SHOULD allow configuring if the Prefix Information option must be carried with the DIO message to distribute the Prefix Information for autoconfiguration. In that case, the RPL implementation MUST allow the list of prefixes to be advertised in the PIO along with the corresponding flags.

- Solicited Information [DIS message, in Solicited Information option]. Note that a RPL implementation SHOULD allow configuring when such messages should be sent and under which circumstances, along with the value of the RPLInstance ID, 'V'/'I'/'D' flags.

- 'K' flag: when a node should set the 'K' flag in a DAO message [DAO message, in DAO Base message].

- MOP (Mode of Operation) [DIO message, in DIO Base message].

- Route Information (and preference) [DIO message, in Route Information option]

18.2.4. Protocol Parameters to Be Configured on Every Non-DODAG-Root Router in the LLN

A RPL implementation MUST allow configuring the Target prefix [DAO message, in RPL Target option].

Furthermore, there are circumstances where a node may want to designate a Target to allow for specific processing of the Target (prioritization, etc.). Such processing rules are out of scope for this specification. When used, a RPL implementation SHOULD allow configuring the Target Descriptor on a per-Target basis (for example, using access lists).

A node whose DODAG parent set is empty may become the DODAG root of a floating DODAG. It may also set its DAGPreference such that it is less preferred. Thus, a RPL implementation MUST allow configuring the set of actions that the node should initiate in this case:

- Start its own (floating) DODAG: the new DODAGID must be configured in addition to its DAGPreference.
- Poison the broken path (see procedure in Section 8.2.2.5).
- Trigger a local repair.

18.2.5. Parameters to Be Configured on the DODAG Root

In addition, several other parameters are configured only on the DODAG root and advertised in options carried in DIO messages.

As specified in Section 8.3, a RPL implementation makes use of Trickle timers to govern the sending of DIO messages. The operation of the Trickle algorithm is determined by a set of configurable parameters, which MUST be configurable and that are then advertised by the DODAG root along the DODAG in DIO messages.

- DIOIntervalDoublings [DIO message, in DODAG Configuration option]
- DIOIntervalMin [DIO message, in DODAG Configuration option]
- DIORedundancyConstant [DIO message, in DODAG Configuration option]

In addition, a RPL implementation SHOULD allow for configuring the following set of RPL parameters:

- Path Control Size [DIO message, in DODAG Configuration option]
- MinHopRankIncrease [DIO message, in DODAG Configuration option]
- The DODAGPreference field [DIO message, DIO Base object]
- DODAGID [DIO message, in DIO Base option] and [DAO message, when the 'D' flag of the DAO message is set]

DAG root behavior: in some cases, a node may not want to permanently act as a floating DODAG root if it cannot join a grounded DODAG. For example, a battery-operated node may not want to act as a floating DODAG root for a long period of time. Thus, a RPL implementation MAY support the ability to configure whether or not a node could act as a floating DODAG root for a configured period of time.

DAG Version Number Increment: a RPL implementation may allow, by configuration at the DODAG root, refreshing the DODAG states by updating the DODAGVersionNumber. A RPL implementation SHOULD allow configuring whether or not periodic or event triggered mechanisms are used by the DODAG root to control DODAGVersionNumber change (which triggers a global repair as specified in Section 3.2.2).

18.2.6. Configuration of RPL Parameters Related to DAO-Based Mechanisms

DAO messages are optional and used in DODAGs that require Downward routing operation. This section deals with the set of parameters related to DAO messages and provides recommendations on their configuration.

As stated in Section 9.5, it is recommended to delay the sending of DAO message to DAO parents in order to maximize the chances to perform route aggregation. Upon receiving a DAO message, the node should thus start a DelayDAO timer. The default value is DEFAULT_DAO_DELAY. A RPL implementation MAY allow for configuring the DelayDAO timer.

In a Storing mode of operation, a storing node may increment DTSN in order to reliably trigger a set of DAO updates from its immediate children, as part of routine routing table updates and maintenance. A RPL implementation MAY allow for configuring a set of rules specifying the triggers for DTSN increment (manual or event-based).

When a DAO entry times out or is invalidated, a node SHOULD make a reasonable attempt to report a No-Path to each of the DAO parents. That number of attempts MAY be configurable.

An implementation should support rate-limiting the sending of DAO messages. The related parameters MAY be configurable.

18.2.7. Configuration of RPL Parameters Related to Security Mechanisms

As described in Section 10, the security features described in this document are optional to implement and a given implementation may support a subset (including the empty set) of the described security features.

To this end, an implementation supporting described security features may conceptually implement a security policy database. In support of the security mechanisms, a RPL implementation SHOULD allow for configuring a subset of the following parameters:

- Security Modes accepted [Unsecured mode, Preinstalled mode, Authenticated mode]
- KIM values accepted [Secure RPL control messages, in Security section]
- Level values accepted [Secure RPL control messages, in Security section]
- Algorithm values accepted [Secure RPL control messages, in Security section]
- Key material in support of Authenticated or Preinstalled key modes.

In addition, a RPL implementation SHOULD allow for configuring a DODAG root with a subset of the following parameters:

- Level values advertised [Secure DIO message, in Security section]
- KIM value advertised [Secure DIO message, in Security section]
- Algorithm value advertised [Secure DIO message, in Security section]

18.2.8. Default Values

This document specifies default values for the following set of RPL variables:

DEFAULT_PATH_CONTROL_SIZE
DEFAULT_DIO_INTERVAL_MIN
DEFAULT_DIO_INTERVAL_DOUBLINGS
DEFAULT_DIO_REDUNDANCY_CONSTANT

DEFAULT_MIN_HOP_RANK_INCREASE
DEFAULT_DAO_DELAY

It is recommended to specify default values in protocols; that being said, as discussed in [RFC5706], default values may make less and less sense. RPL is a routing protocol that is expected to be used in a number of contexts where network characteristics such as the number of nodes and link and node types are expected to vary significantly. Thus, these default values are likely to change with the context and as the technology evolves. Indeed, LLNs' related technology (e.g., hardware, link layers) have been evolving dramatically over the past few years and such technologies are expected to change and evolve considerably in the coming years.

The proposed values are not based on extensive best current practices and are considered to be conservative.

18.3. Monitoring of RPL Operation

Several RPL parameters should be monitored to verify the correct operation of the routing protocol and the network itself. This section lists the set of monitoring parameters of interest.

18.3.1. Monitoring a DODAG Parameters

A RPL implementation SHOULD provide information about the following parameters:

- DODAG Version number [DIO message, in DIO Base message]
- Status of the 'G' flag [DIO message, in DIO Base message]
- Status of the MOP field [DIO message, in DIO Base message]
- Value of the DTSN [DIO message, in DIO Base message]
- Value of the Rank [DIO message, in DIO Base message]
- DAOSequence: Incremented at each unique DAO message, echoed in the DAO-ACK message [DAO and DAO-ACK messages]
- Route Information [DIO message, Route Information Option] (list of IPv6 prefixes per parent along with lifetime and preference]
- Trickle parameters:
  - DIOIntervalDoublings [DIO message, in DODAG Configuration option]

* DIOIntervalMin [DIO message, in DODAG Configuration option]

* DIORedundancyConstant [DIO message, in DODAG Configuration option]

- Path Control Size [DIO message, in DODAG Configuration option]

- MinHopRankIncrease [DIO message, in DODAG Configuration option]

Values that may be monitored only on the DODAG root:

- Transit Information [DAO, Transit Information option]: A RPL implementation SHOULD allow configuring whether the set of received Transit Information options should be displayed on the DODAG root. In this case, the RPL database of received Transit Information should also contain the Path Sequence, Path Control, Path Lifetime, and Parent Address.

18.3.2. Monitoring a DODAG Inconsistencies and Loop Detection

Detection of DODAG inconsistencies is particularly critical in RPL networks. Thus, it is recommended for a RPL implementation to provide appropriate monitoring tools. A RPL implementation SHOULD provide a counter reporting the number of a times the node has detected an inconsistency with respect to a DODAG parent, e.g., if the DODAGID has changed.

When possible more granular information about inconsistency detection should be provided. A RPL implementation MAY provide counters reporting the number of following inconsistencies:

- Packets received with 'O' bit set (to Down) from a node with a higher Rank

- Packets received with 'O' bit cleared (to Up) from a node with a lower Rank

- Number of packets with the 'F' bit set

- Number of packets with the 'R' bit set

18.4. Monitoring of the RPL Data Structures

18.4.1. Candidate Neighbor Data Structure

A node in the candidate neighbor list is a node discovered by the same means and qualified to potentially become a parent (with high enough local confidence). A RPL implementation SHOULD provide a way

to allow for the candidate neighbor list to be monitored with some metric reflecting local confidence (the degree of stability of the neighbors) as measured by some metrics.

A RPL implementation MAY provide a counter reporting the number of times a candidate neighbor has been ignored, should the number of candidate neighbors exceed the maximum authorized value.

18.4.2. Destination-Oriented Directed Acyclic Graph (DODAG) Table

For each DODAG, a RPL implementation is expected to keep track of the following DODAG table values:

- RPLInstanceID
- DODAGID
- DODAGVersionNumber
- Rank
- Objective Code Point
- A set of DODAG parents
- A set of prefixes offered Upward along the DODAG
- Trickle timers used to govern the sending of DIO messages for the DODAG
- List of DAO parents
- DTSN
- Node status (router versus leaf)

A RPL implementation SHOULD allow for monitoring the set of parameters listed above.

18.4.3. Routing Table and DAO Routing Entries

A RPL implementation maintains several information elements related to the DODAG and the DAO entries (for storing nodes). In the case of a non-storing node, a limited amount of information is maintained (the routing table is mostly reduced to a set of DODAG parents along with characteristics of the DODAG as mentioned above); whereas in the case of storing nodes, this information is augmented with routing entries.

A RPL implementation SHOULD allow for the following parameters to be monitored:

- Next Hop (DODAG parent)
- Next Hop Interface
- Path metrics value for each DODAG parent

A DAO Routing Table entry conceptually contains the following elements (for storing nodes only):

- Advertising Neighbor Information
- IPv6 address
- Interface ID to which DAO parents has this entry been reported
- Retry counter
- Logical equivalent of DAO Content:
  - DAO-Sequence
  - Path Sequence
  - DAO Lifetime
  - DAO Path Control
- Destination Prefix (or address or Mcast Group)

A RPL implementation SHOULD provide information about the state of each DAO Routing Table entry states.

18.5. Fault Management

Fault management is a critical component used for troubleshooting, verification of the correct mode of operation of the protocol, and network design; also, it is a key component of network performance monitoring. A RPL implementation SHOULD allow the provision of the following information related to fault managements:

- Memory overflow along with the cause (e.g., routing tables overflow, etc.)
- Number of times a packet could not be sent to a DODAG parent flagged as valid

- Number of times a packet has been received for which the router did not have a corresponding RPLInstanceID
- Number of times a local repair procedure was triggered
- Number of times a global repair was triggered by the DODAG root
- Number of received malformed messages
- Number of seconds with packets to forward and no next hop (DODAG parent)
- Number of seconds without next hop (DODAG parent)
- Number of times a node has joined a DODAG as a leaf because it received a DIO with a metric/constraint that was not understood and it was configured to join as a leaf node in this case (see Section 18.6)

It is RECOMMENDED to report faults via at least error log messages. Other protocols may be used to report such faults.

18.6. Policy

Policy rules can be used by a RPL implementation to determine whether or not the node is allowed to join a particular DODAG advertised by a neighbor by means of DIO messages.

This document specifies operation within a single DODAG. A DODAG is characterized by the following tuple (RPLInstanceID, DODAGID). Furthermore, as pointed out above, DIO messages are used to advertise other DODAG characteristics such as the routing metrics and constraints used to build to the DODAG and the Objective Function in use (specified by OCP).

The first policy rules consist of specifying the following conditions that a RPL node must satisfy to join a DODAG:

- RPLInstanceID
- List of supported routing metrics and constraints
- Objective Function (OCP values)

A RPL implementation MUST allow configuring these parameters and SHOULD specify whether the node must simply ignore the DIO if the advertised DODAG is not compliant with the local policy or whether the node should join as the leaf node if only the list of supported

routing metrics and constraints, and the OF is not supported. Additionally, a RPL implementation SHOULD allow for the addition of the DODAGID as part of the policy.

A RPL implementation SHOULD allow configuring the set of acceptable or preferred Objective Functions (OFs) referenced by their Objective Code Points (OCPs) for a node to join a DODAG, and what action should be taken if none of a node's candidate neighbors advertise one of the configured allowable Objective Functions, or if the advertised metrics/constraint is not understood/supported. Two actions can be taken in this case:

- The node joins the DODAG as a leaf node as specified in Section 8.5.
- The node does not join the DODAG.

A node in an LLN may learn routing information from different routing protocols including RPL. In this case, it is desirable to control, via administrative preference, which route should be favored. An implementation SHOULD allow for the specification of an administrative preference for the routing protocol from which the route was learned.

Internal Data Structures: some RPL implementations may limit the size of the candidate neighbor list in order to bound the memory usage; in which case, some otherwise viable candidate neighbors may not be considered and simply dropped from the candidate neighbor list.

A RPL implementation MAY provide an indicator on the size of the candidate neighbor list.

18.7. Fault Isolation

It is RECOMMENDED to quarantine neighbors that start emitting malformed messages at unacceptable rates.

18.8. Impact on Other Protocols

RPL has very limited impact on other protocols. Where more than one routing protocol is required on a router, such as an LBR, it is expected for the device to support routing redistribution functions between the routing protocols to allow for reachability between the two routing domains. Such redistribution SHOULD be governed by the use of user configurable policy.

With regard to the impact in terms of traffic on the network, RPL has been designed to limit the control traffic thanks to mechanisms such as Trickle timers (Section 8.3). Thus, the impact of RPL on other protocols should be extremely limited.

18.9. Performance Management

Performance management is always an important aspect of a protocol, and RPL is not an exception. Several metrics of interest have been specified by the IP Performance Monitoring (IPPM) working group: that being said, they will be hardly applicable to LLN considering the cost of monitoring these metrics in terms of resources on the devices and required bandwidth. Still, RPL implementations MAY support some of these, and other parameters of interest are listed below:

- Number of repairs and time to repair in seconds (average, variance)
- Number of times and time period during which a devices could not forward a packet because of a lack of a reachable neighbor in its routing table
- Monitoring of resources consumption by RPL in terms of bandwidth and required memory
- Number of RPL control messages sent and received

18.10. Diagnostics

There may be situations where a node should be placed in "verbose" mode to improve diagnostics. Thus, a RPL implementation SHOULD provide the ability to place a node in and out of verbose mode in order to get additional diagnostic information.

19. Security Considerations

19.1. Overview

From a security perspective, RPL networks are no different from any other network. They are vulnerable to passive eavesdropping attacks and, potentially, even active tampering when physical access to a wire is not required to participate in communications. The very nature of ad hoc networks and their cost objectives impose additional security constraints, which perhaps make these networks the most difficult environments to secure. Devices are low-cost and have limited capabilities in terms of computing power, available storage, and power drain; it cannot always be assumed they have a trusted computing base or a high-quality random number generator aboard.

Communications cannot rely on the online availability of a fixed infrastructure and might involve short-term relationships between devices that may never have communicated before. These constraints might severely limit the choice of cryptographic algorithms and protocols and influence the design of the security architecture because the establishment and maintenance of trust relationships between devices need to be addressed with care. In addition, battery lifetime and cost constraints put severe limits on the security overhead these networks can tolerate, something that is of far less concern with higher bandwidth networks. Most of these security architectural elements can be implemented at higher layers and may, therefore, be considered to be out of scope for this specification. Special care, however, needs to be exercised with respect to interfaces to these higher layers.

The security mechanisms in this standard are based on symmetric-key and public-key cryptography and use keys that are to be provided by higher-layer processes. The establishment and maintenance of these keys are out of scope for this specification. The mechanisms assume a secure implementation of cryptographic operations and secure and authentic storage of keying material.

The security mechanisms specified provide particular combinations of the following security services:

Data confidentiality: Assurance that transmitted information is only disclosed to parties for which it is intended.

Data authenticity: Assurance of the source of transmitted information (and, hereby, that information was not modified in transit).

Replay protection: Assurance that a duplicate of transmitted information is detected.

Timeliness (delay protection): Assurance that transmitted information was received in a timely manner.

The actual protection provided can be adapted on a per-packet basis and allows for varying levels of data authenticity (to minimize security overhead in transmitted packets where required) and for optional data confidentiality. When nontrivial protection is required, replay protection is always provided.

Replay protection is provided via the use of a non-repeating value (CCM nonce) in the packet protection process and storage of some status information (originating device and the CCM nonce counter last received from that device), which allows detection of whether this particular CCM nonce value was used previously by the originating

device. In addition, so-called delay protection is provided amongst those devices that have a loosely synchronized clock on board. The acceptable time delay can be adapted on a per-packet basis and allows for varying latencies (to facilitate longer latencies in packets transmitted over a multi-hop communication path).

Cryptographic protection may use a key shared between two peer devices (link key) or a key shared among a group of devices (group key), thus allowing some flexibility and application-specific trade-offs between key storage and key maintenance costs versus the cryptographic protection provided. If a group key is used for peer-to-peer communication, protection is provided only against outsider devices and not against potential malicious devices in the key-sharing group.

Data authenticity may be provided using symmetric-key-based or public-key-based techniques. With public-key-based techniques (via signatures), one corroborates evidence as to the unique originator of transmitted information, whereas with symmetric-key-based techniques, data authenticity is only provided relative to devices in a key-sharing group. Thus, public-key-based authentication may be useful in scenarios that require a more fine-grained authentication than can be provided with symmetric-key-based authentication techniques alone, such as with group communications (broadcast, multicast) or in scenarios that require non-repudiation.

20. IANA Considerations

20.1. RPL Control Message

The RPL control message is an ICMP information message type that is to be used carry DODAG Information Objects, DODAG Information Solicitations, and Destination Advertisement Objects in support of RPL operation.

IANA has defined an ICMPv6 Type Number Registry. The type value for the RPL control message is 155.

20.2. New Registry for RPL Control Codes

IANA has created a registry, RPL Control Codes, for the Code field of the ICMPv6 RPL control message.

New codes may be allocated only by an IETF Review. Each code is tracked with the following qualities:

- Code

- Description
- Defining RFC

The following codes are currently defined:

Code	Description	Reference
0x00	DODAG Information Solicitation	This document
0x01	DODAG Information Object	This document
0x02	Destination Advertisement Object	This document
0x03	Destination Advertisement Object Acknowledgment	This document
0x80	Secure DODAG Information Solicitation	This document
0x81	Secure DODAG Information Object	This document
0x82	Secure Destination Advertisement Object	This document
0x83	Secure Destination Advertisement Object Acknowledgment	This document
0x8A	Consistency Check	This document

RPL Control Codes

20.3. New Registry for the Mode of Operation (MOP)

IANA has created a registry for the 3-bit Mode of Operation (MOP), which is contained in the DIO Base.

New values may be allocated only by an IETF Review. Each value is tracked with the following qualities:

- Mode of Operation Value

- Capability description
- Defining RFC

Four values are currently defined:

MOP value	Description	Reference
0	No Downward routes maintained by RPL	This document
1	Non-Storing Mode of Operation	This document
2	Storing Mode of Operation with no multicast support	This document
3	Storing Mode of Operation with multicast support	This document

DIO Mode of Operation

The rest of the range, decimal 4 to 7, is currently unassigned.

20.4. RPL Control Message Options

IANA has created a registry for the RPL Control Message Options.

New values may be allocated only by an IETF Review. Each value is tracked with the following qualities:

- Value
- Meaning
- Defining RFC

Value	Meaning	Reference
0x00	Pad1	This document
0x01	PadN	This document
0x02	DAG Metric Container	This Document
0x03	Routing Information	This Document
0x04	DODAG Configuration	This Document
0x05	RPL Target	This Document
0x06	Transit Information	This Document
0x07	Solicited Information	This Document
0x08	Prefix Information	This Document
0x09	Target Descriptor	This Document

RPL Control Message Options

20.5. Objective Code Point (OCP) Registry

IANA has created a registry to manage the codespace of the Objective Code Point (OCP) field.

No OCPs are defined in this specification.

New codes may be allocated only by an IETF Review. Each code is tracked with the following qualities:

- Code
- Description
- Defining RFC

20.6. New Registry for the Security Section Algorithm

IANA has created a registry for the values of the 8-bit Algorithm field in the Security section.

New values may be allocated only by an IETF Review. Each value is tracked with the following qualities:

- Value
- Encryption/MAC
- Signature
- Defining RFC

The following value is currently defined:

Value	Encryption/MAC	Signature	Reference
0	CCM with AES-128	RSA with SHA-256	This document

Security Section Algorithm

20.7. New Registry for the Security Section Flags

IANA has created a registry for the 8-bit Security Section Flags field.

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- Bit number (counting from bit 0 as the most significant bit)
- Capability description
- Defining RFC

No bit is currently defined for the Security Section Flags field.

20.8. New Registry for Per-KIM Security Levels

IANA has created one registry for the 3-bit Security Level (LVL) field per allocated KIM value.

For a given KIM value, new levels may be allocated only by an IETF Review. Each level is tracked with the following qualities:

- Level
- KIM value

- Description
- Defining RFC

The following levels per KIM value are currently defined:

Level	KIM value	Description	Reference
0	0	See Figure 11	This document
1	0	See Figure 11	This document
2	0	See Figure 11	This document
3	0	See Figure 11	This document
0	1	See Figure 11	This document
1	1	See Figure 11	This document
2	1	See Figure 11	This document
3	1	See Figure 11	This document
0	2	See Figure 11	This document
1	2	See Figure 11	This document
2	2	See Figure 11	This document
3	2	See Figure 11	This document
0	3	See Figure 11	This document
1	3	See Figure 11	This document
2	3	See Figure 11	This document
3	3	See Figure 11	This document

Per-KIM Security Levels

20.9. New Registry for DODAG Informational Solicitation (DIS) Flags

IANA has created a registry for the DIS (DODAG Informational Solicitation) Flags field.

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- Bit number (counting from bit 0 as the most significant bit)
- Capability description
- Defining RFC

No bit is currently defined for the DIS (DODAG Informational Solicitation) Flags field.

20.10. New Registry for the DODAG Information Object (DIO) Flags

IANA has created a registry for the 8-bit DODAG Information Object (DIO) Flags field.

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- Bit number (counting from bit 0 as the most significant bit)
- Capability description
- Defining RFC

No bit is currently defined for the DIS (DODAG Informational Solicitation) Flags.

20.11. New Registry for the Destination Advertisement Object (DAO) Flags

IANA has created a registry for the 8-bit Destination Advertisement Object (DAO) Flags field.

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- Bit number (counting from bit 0 as the most significant bit)
- Capability description
- Defining RFC

The following bits are currently defined:

Bit number	Description	Reference
0	DAO-ACK request (K)	This document
1	DODAGID field is present (D)	This document

DAO Base Flags

20.12. New Registry for the Destination Advertisement Object (DAO) Acknowledgement Flags

IANA has created a registry for the 8-bit Destination Advertisement Object (DAO) Acknowledgement Flags field.

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- Bit number (counting from bit 0 as the most significant bit)
- Capability description
- Defining RFC

The following bit is currently defined:

Bit number	Description	Reference
0	DODAGID field is present (D)	This document

DAO-ACK Base Flags

20.13. New Registry for the Consistency Check (CC) Flags

IANA has created a registry for the 8-bit Consistency Check (CC) Flags field.

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- Bit number (counting from bit 0 as the most significant bit)
- Capability description

- Defining RFC

The following bit is currently defined:

Bit number	Description	Reference
0	CC Response (R)	This document

Consistency Check Base Flags

20.14. New Registry for the DODAG Configuration Option Flags

IANA has created a registry for the 8-bit DODAG Configuration Option Flags field.

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- Bit number (counting from bit 0 as the most significant bit)
- Capability description
- Defining RFC

The following bits are currently defined:

Bit number	Description	Reference
4	Authentication Enabled (A)	This document
5-7	Path Control Size (PCS)	This document

DODAG Configuration Option Flags

20.15. New Registry for the RPL Target Option Flags

IANA has created a registry for the 8-bit RPL Target Option Flags field.

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- Bit number (counting from bit 0 as the most significant bit)
- Capability description

- Defining RFC

No bit is currently defined for the RPL Target Option Flags.

20.16. New Registry for the Transit Information Option Flags

IANA has created a registry for the 8-bit Transit Information Option (TIO) Flags field.

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- Bit number (counting from bit 0 as the most significant bit)
- Capability description
- Defining RFC

The following bits are currently defined:

Bit number	Description	Reference
0	External (E)	This document

Transit Information Option Flags

20.17. New Registry for the Solicited Information Option Flags

IANA has created a registry for the 8-bit Solicited Information Option (SIO) Flags field.

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- Bit number (counting from bit 0 as the most significant bit)
- Capability description
- Defining RFC

The following bits are currently defined:

Bit number	Description	Reference
0	Version Predicate match (V)	This document
1	InstanceID Predicate match (I)	This document
2	DODAGID Predicate match (D)	This document

Solicited Information Option Flags

20.18. ICMPv6: Error in Source Routing Header

In some cases RPL will return an ICMPv6 error message when a message cannot be delivered as specified by its source routing header. This ICMPv6 error message is "Error in Source Routing Header".

IANA has defined an ICMPv6 "Code" Fields Registry for ICMPv6 Message Types. ICMPv6 Message Type 1 describes "Destination Unreachable" codes. The "Error in Source Routing Header" code is has been allocated from the ICMPv6 Code Fields Registry for ICMPv6 Message Type 1, with a code value of 7.

20.19. Link-Local Scope Multicast Address

The rules for assigning new IPv6 multicast addresses are defined in [RFC3307]. This specification requires the allocation of a new permanent multicast address with a link-local scope for RPL nodes called all-RPL-nodes, with a value of ff02::1a.

21. Acknowledgements

The authors would like to acknowledge the review, feedback, and comments from Emmanuel Baccelli, Dominique Barthel, Yusuf Bashir, Yoav Ben-Yehezkel, Phoebus Chen, Quynh Dang, Mischa Dohler, Mathilde Durvy, Joakim Eriksson, Omprakash Gnawali, Manhar Goindi, Mukul Goyal, Ulrich Herberg, Anders Jagd, JeongGil (John) Ko, Ajay Kumar, Quentin Lampin, Jerry Martocci, Matteo Paris, Alexandru Petrescu, Joseph Reddy, Michael Richardson, Don Sturek, Joydeep Tripathi, and Nicolas Tsiftes.

The authors would like to acknowledge the guidance and input provided by the ROLL Chairs, David Culler and JP. Vasseur, and the Area Director, Adrian Farrel.

The authors would like to acknowledge prior contributions of Robert Assimiti, Mischa Dohler, Julien Abeille, Ryuji Wakikawa, Teco Boot, Patrick Wetterwald, Bryan Mclaughlin, Carlos J. Bernardos, Thomas Watteyne, Zach Shelby, Caroline Bontoux, Marco Molteni, Billy Moon, Jim Bound, Yanick Pouffary, Henning Rogge, and Arsalan Tavakoli, who have provided useful design considerations to RPL.

RPL Security Design, found in Section 10, Section 19, and elsewhere throughout the document, is primarily the contribution of the Security Design Team: Tzeta Tsao, Roger Alexander, Dave Ward, Philip Levis, Kris Pister, Rene Struik, and Adrian Farrel.

Thanks also to Jari Arkko and Ralph Droms for their attentive reviews, especially with respect to interoperability considerations and integration with other IETF specifications.

22. Contributors

Stephen Dawson-Haggerty
UC Berkeley
Soda Hall
Berkeley, CA 94720
USA

EMail: stevedh@cs.berkeley.edu

23. References

23.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, March 2011.

[RFC6275] Perkins, C., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.

[RFC6551] Vasseur, JP., Ed., Kim, M., Ed., Pister, K., Dejean, N., and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", RFC 6551, March 2012.

[RFC6552] Thubert, P., Ed., "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6552, March 2012.

[RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, March 2012.

[RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, March 2012.

23.2. Informative References

[6LOWPAN-ND] Shelby, Z., Ed., Chakrabarti, S., and E. Nordmark, "Neighbor Discovery Optimization for Low Power and Lossy Networks (6LoWPAN)", Work in Progress, October 2011.

[FIPS180] National Institute of Standards and Technology, "FIPS Pub 180-3, Secure Hash Standard (SHS)", US Department of Commerce , February 2008, <http://www.nist.gov/itl/upload/fips180-3_final.pdf>.

[Perlman83] Perlman, R., "Fault-Tolerant Broadcast of Routing Information", North-Holland Computer Networks, Vol.7: p. 395-405, December 1983.

[RFC1958] Carpenter, B., "Architectural Principles of the Internet", RFC 1958, June 1996.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, August 1996.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC3307] Haberman, B., "Allocation Guidelines for IPv6 Multicast Addresses", RFC 3307, August 2002.

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

[RFC3535] Schoenwaelder, J., "Overview of the 2002 IAB Network Management Workshop", RFC 3535, May 2003.

[RFC3610] Whiting, D., Housley, R., and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3610, September 2003.

[RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.

[RFC4101] Rescorla, E. and IAB, "Writing Protocol Models", RFC 4101, June 2005.

[RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, June 2007.

[RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)", RFC 5120, February 2008.

[RFC5184] Teraoka, F., Gogo, K., Mitsuya, K., Shibui, R., and K. Mitani, "Unified Layer 2 (L2) Abstractions for Layer 3 (L3)-Driven Fast Handover", RFC 5184, May 2008.

[RFC5548] Dohler, M., Watteyne, T., Winter, T., and D. Barthel, "Routing Requirements for Urban Low-Power and Lossy Networks", RFC 5548, May 2009.

[RFC5673] Pister, K., Thubert, P., Dwars, S., and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks", RFC 5673, October 2009.

[RFC5706] Harrington, D., "Guidelines for Considering Operations and Management of New Protocols and Protocol Extensions", RFC 5706, November 2009.

[RFC5826] Brandt, A., Buron, J., and G. Porcu, "Home Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5826, April 2010.

[RFC5867] Martocci, J., De Mil, P., Riou, N., and W. Vermeylen, "Building Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5867, June 2010.

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, June 2010.

[RFC6130] Clausen, T., Dearlove, C., and J. Dean, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)", RFC 6130, April 2011.

[ROLL-TERMS] Vasseur, J., "Terminology in Low power And Lossy Networks", Work in Progress, September 2011.

Appendix A. Example Operation

This appendix provides some examples to illustrate the dissemination of addressing information and prefixes with RPL. The examples depict information being distributed with PIOs and RIOs and the use of DIO and DAO messages. Note that this appendix is not normative, and that the specific details of a RPL addressing plan and autoconfiguration may vary according to specific implementations. RPL merely provides a vehicle for disseminating information that may be built upon and used by other mechanisms.

Note that these examples illustrate use of address autoconfiguration schemes supported by information distributed within RPL. However, if an implementation includes another address autoconfiguration scheme, RPL nodes might be configured not to set the 'A' flag in PIO options, though the PIO can still be used to distribute prefix and addressing information.

A.1. Example Operation in Storing Mode with Node-Owned Prefixes

Figure 32 illustrates the logical addressing architecture of a simple RPL network operating in Storing mode. In this example, each Node, A, B, C, and D, owns its own prefix and makes that prefix available for address autoconfiguration by on-link devices. (This is conveyed by setting the 'A' flag and the 'L' flag in the PIO of the DIO messages). Node A owns the prefix A::/64, Node B owns B::/64, and so on. Node B autoconfigures an on-link address with respect to Node A, A::B. Nodes C and D similarly autoconfigure on-link addresses from Node B's prefix, B::C and B::D, respectively. Nodes have the option of setting the 'R' flag and publishing their address within the Prefix field of the PIO.

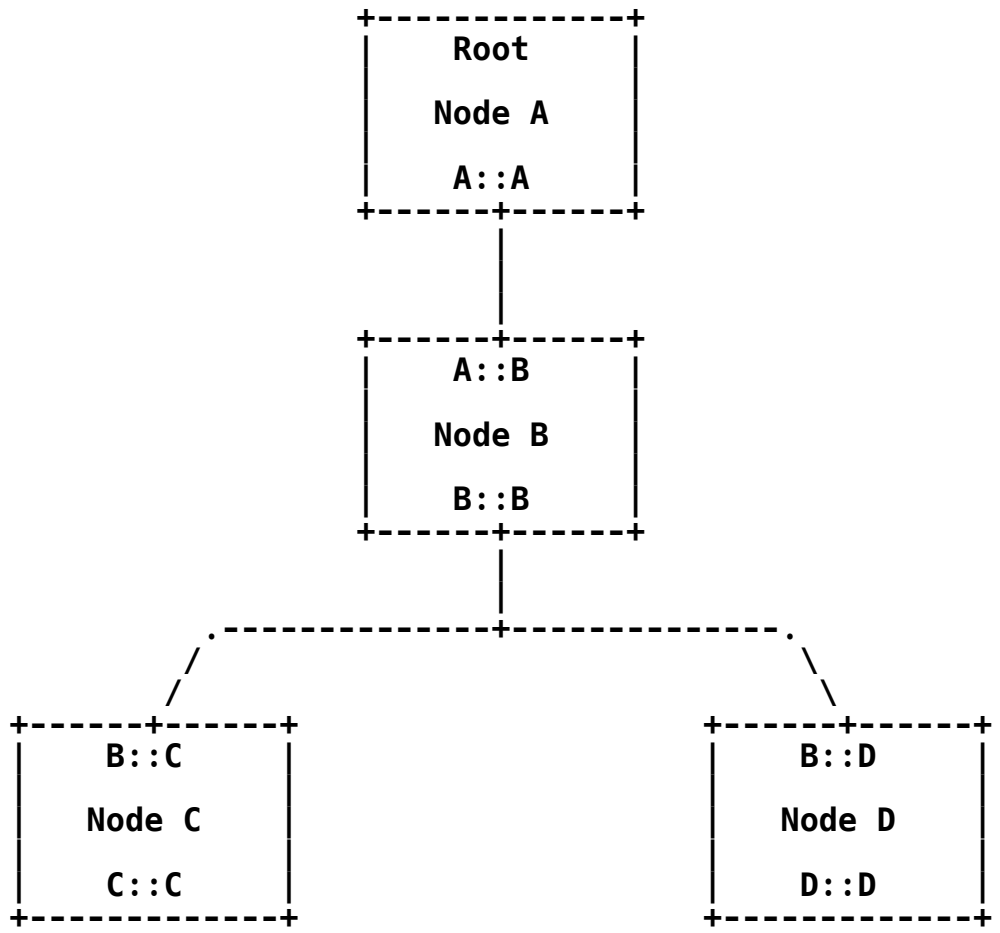


Figure 32: Storing Mode with Node-Owned Prefixes

A.1.1. DIO Messages and PIO

Node A, for example, will send DIO messages with a PIO as follows:

'A' flag:	Set
'L' flag:	Set
'R' flag:	Clear
Prefix Length:	64
Prefix:	A::

Node B, for example, will send DIO messages with a PIO as follows:

'A' flag:	Set
'L' flag:	Set
'R' flag:	Set
Prefix Length:	64
Prefix:	B::B

Node C, for example, will send DIO messages with a PIO as follows:

```
'A' flag:       Set
'L' flag:       Set
'R' flag:       Clear
Prefix Length:  64
Prefix:         C::
```

Node D, for example, will send DIO messages with a PIO as follows:

```
'A' flag:       Set
'L' flag:       Set
'R' flag:       Set
Prefix Length:  64
Prefix:         D::D
```

A.1.2. DAO Messages

Node B will send DAO messages to Node A with the following information:

- Target B::/64
- Target C::/64
- Target D::/64

Node C will send DAO messages to Node B with the following information:

- Target C::/64

Node D will send DAO messages to Node B with the following information:

- Target D::/64

A.1.3. Routing Information Base

Node A will conceptually collect the following information into its Routing Information Base (RIB):

- A::/64 connected
- B::/64 via B's link local
- C::/64 via B's link local
- D::/64 via B's link local

Node B will conceptually collect the following information into its RIB:

- ::/0 via A's link local
- B::/64 connected
- C::/64 via C's link local
- D::/64 via D's link local

Node C will conceptually collect the following information into its RIB:

- o ::/0 via B's link local
- o C::/64 connected

Node D will conceptually collect the following information into its RIB:

- o ::/0 via B's link local
- o D::/64 connected

A.2. Example Operation in Storing Mode with Subnet-Wide Prefix

Figure 33 illustrates the logical addressing architecture of a simple RPL network operating in Storing mode. In this example, the root Node A sources a prefix that is used for address autoconfiguration over the entire RPL subnet. (This is conveyed by setting the 'A' flag and clearing the 'L' flag in the PIO of the DIO messages.) Nodes A, B, C, and D all autoconfigure to the prefix A::/64. Nodes have the option of setting the 'R' flag and publishing their address within the Prefix field of the PIO.

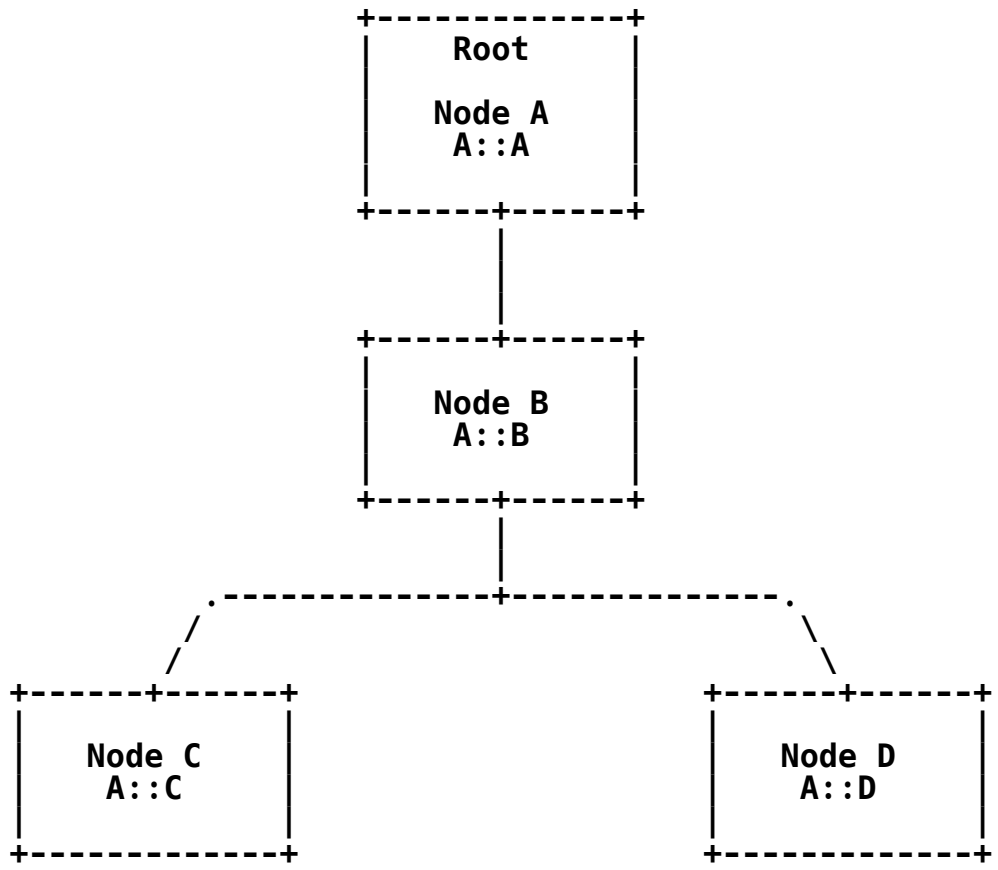


Figure 33: Storing Mode with Subnet-Wide Prefix

A.2.1. DIO Messages and PIO

Node A, for example, will send DIO messages with a PIO as follows:

'A' flag:	Set
'L' flag:	Clear
'R' flag:	Clear
Prefix Length:	64
Prefix:	A::

Node B, for example, will send DIO messages with a PIO as follows:

'A' flag:	Set
'L' flag:	Clear
'R' flag:	Set
Prefix Length:	64
Prefix:	A::B

Node C, for example, will send DIO messages with a PIO as follows:

```
'A' flag:       Set
'L' flag:       Clear
'R' flag:       Clear
Prefix Length:  64
Prefix:         A::
```

Node D, for example, will send DIO messages with a PIO as follows:

```
'A' flag:       Set
'L' flag:       Clear
'R' flag:       Set
Prefix Length:  64
Prefix:         A::D
```

A.2.2. DAO Messages

Node B will send DAO messages to Node A with the following information:

- Target A::B/128
- Target A::C/128
- Target A::D/128

Node C will send DAO messages to Node B with the following information:

- Target A::C/128

Node D will send DAO messages to Node B with the following information:

- Target A::D/128

A.2.3. Routing Information Base

Node A will conceptually collect the following information into its RIB:

- A::A/128 connected
- A::B/128 via B's link local
- A::C/128 via B's link local
- A::D/128 via B's link local

Node B will conceptually collect the following information into its RIB:

- ::/0 via A's link local
- A::B/128 connected
- A::C/128 via C's link local
- A::D/128 via D's link local

Node C will conceptually collect the following information into its RIB:

- ::/0 via B's link local
- A::C/128 connected

Node D will conceptually collect the following information into its RIB:

- ::/0 via B's link local
- A::D/128 connected

A.3. Example Operation in Non-Storing Mode with Node-Owned Prefixes

Figure 34 illustrates the logical addressing architecture of a simple RPL network operating in Non-Storing mode. In this example, each Node, A, B, C, and D, owns its own prefix, and makes that prefix available for address autoconfiguration by on-link devices. (This is conveyed by setting the 'A' flag and the 'L' flag in the PIO of the DIO messages.) Node A owns the prefix A::/64, Node B owns B::/64, and so on. Node B autoconfigures an on-link address with respect to Node A, A::B. Nodes C and D similarly autoconfigure on-link addresses from Node B's prefix, B::C and B::D, respectively. Nodes have the option of setting the 'R' flag and publishing their address within the Prefix field of the PIO.

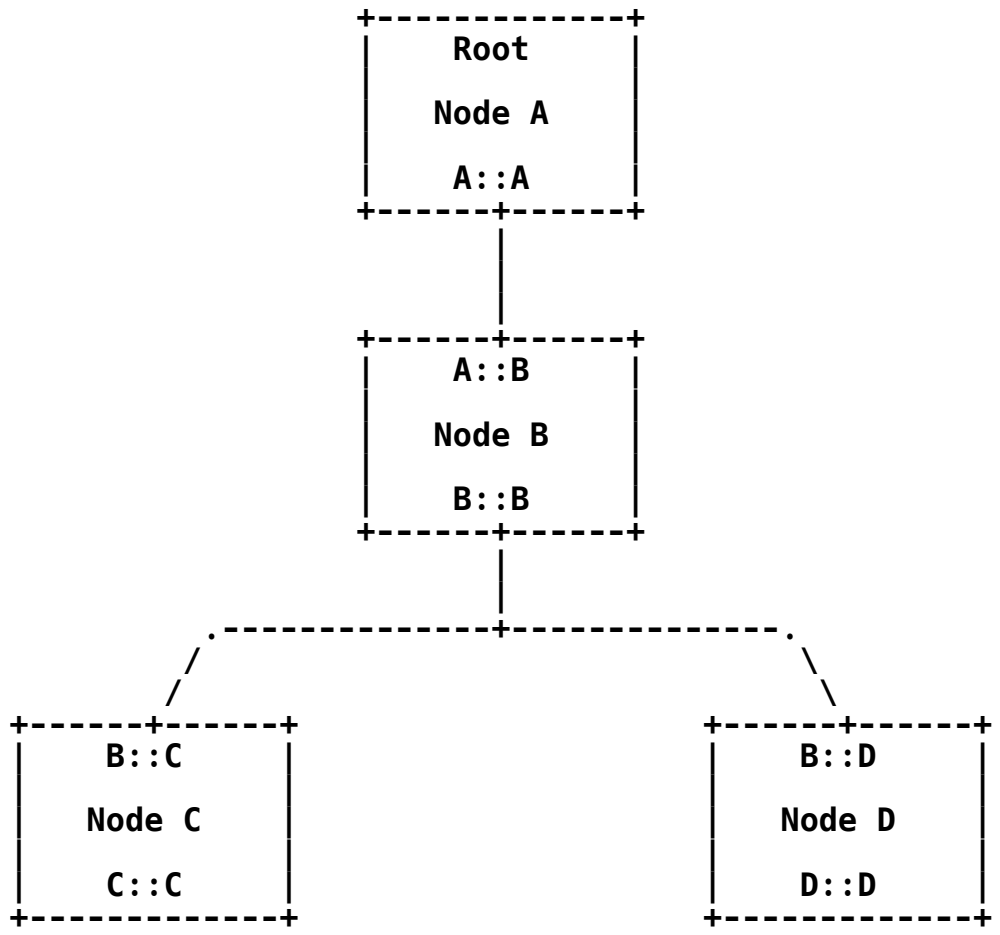


Figure 34: Non-Storing Mode with Node-Owned Prefixes

A.3.1. DIO Messages and PIO

The PIO contained in the DIO messages in the Non-Storing mode with node-owned prefixes can be considered to be identical to those in the Storing mode with node-owned prefixes case (Appendix A.1.1).

A.3.2. DAO Messages

Node B will send DAO messages to Node A with the following information:

- Target B::/64, Transit A::B

Node C will send DAO messages to Node A with the following information:

- Target C::/64, Transit B::C

Node D will send DAO messages to Node A with the following information:

- Target D::/64, Transit B::D

A.3.3. Routing Information Base

Node A will conceptually collect the following information into its RIB. Note that Node A has enough information to construct source routes by doing recursive lookups into the RIB:

- A::/64 connected
- B::/64 via A::B
- C::/64 via B::C
- D::/64 via B::D

Node B will conceptually collect the following information into its RIB:

- ::/0 via A's link local
- B::/64 connected

Node C will conceptually collect the following information into its RIB:

- ::/0 via B's link local
- C::/64 connected

Node D will conceptually collect the following information into its RIB:

- ::/0 via B's link local
- D::/64 connected

A.4. Example Operation in Non-Storing Mode with Subnet-Wide Prefix

Figure 35 illustrates the logical addressing architecture of a simple RPL network operating in Non-Storing mode. In this example, the root Node A sources a prefix that is used for address autoconfiguration over the entire RPL subnet. (This is conveyed by setting the 'A' flag and clearing the 'L' flag in the PIO of the DIO messages.) Nodes A, B, C, and D all autoconfigure to the prefix A::/64. Nodes must set the 'R' flag and publish their address within the Prefix field of the PIO, in order to inform their children which address to use in the transit option.

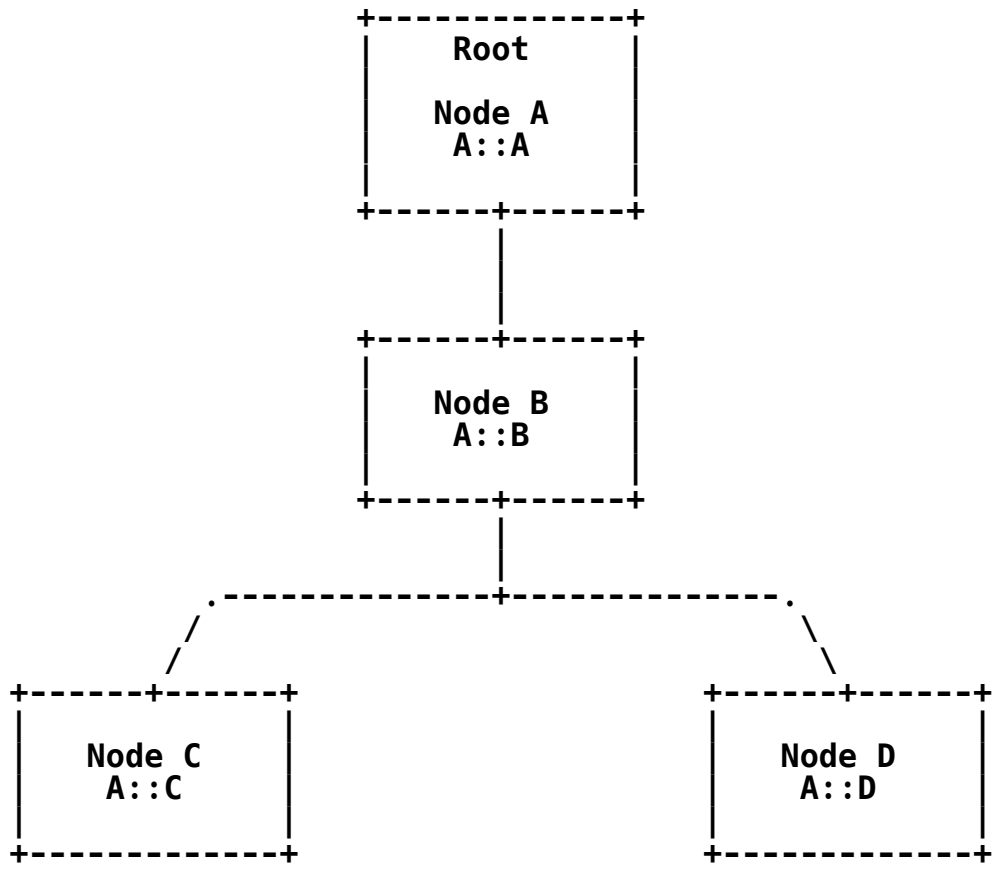


Figure 35: Non-Storing Mode with Subnet-Wide Prefix

A.4.1. DIO Messages and PIO

Node A, for example, will send DIO messages with a PIO as follows:

'A' flag:	Set
'L' flag:	Clear
'R' flag:	Set
Prefix Length:	64
Prefix:	A::A

Node B, for example, will send DIO messages with a PIO as follows:

'A' flag:	Set
'L' flag:	Clear
'R' flag:	Set
Prefix Length:	64
Prefix:	A::B

Node C, for example, will send DIO messages with a PIO as follows:
'A' flag: Set
'L' flag: Clear
'R' flag: Set
Prefix Length: 64
Prefix: A::C

Node D, for example, will send DIO messages with a PIO as follows:
'A' flag: Set
'L' flag: Clear
'R' flag: Set
Prefix Length: 64
Prefix: A::D

A.4.2. DAO Messages

Node B will send DAO messages to Node A with the following information:
- Target A::B/128, Transit A::A

Node C will send DAO messages to Node A with the following information:
- Target A::C/128, Transit A::B

Node D will send DAO messages to Node A with the following information:
- Target A::D/128, Transit A::B

A.4.3. Routing Information Base

Node A will conceptually collect the following information into its RIB. Note that Node A has enough information to construct source routes by doing recursive lookups into the RIB:
- A::A/128 connected
- A::B/128 via A::A
- A::C/128 via A::B
- A::D/128 via A::B

Node B will conceptually collect the following information into its RIB:
- ::/0 via A's link local
- A::B/128 connected

Node C will conceptually collect the following information into its RIB:
- ::/0 via B's link local
- A::C/128 connected

Node D will conceptually collect the following information into its RIB:

- o ::/0 via B's link local
- o A::D/128 connected

A.5. Example with External Prefixes

Consider the simple network illustrated in Figure 36. In this example, there are a group of routers participating in a RPL network: a DODAG root, Nodes A, Y, and Z. The DODAG root and Node Z also have connectivity to different external network domains (i.e., external to the RPL network). Note that those external networks could be RPL networks or another type of network altogether.

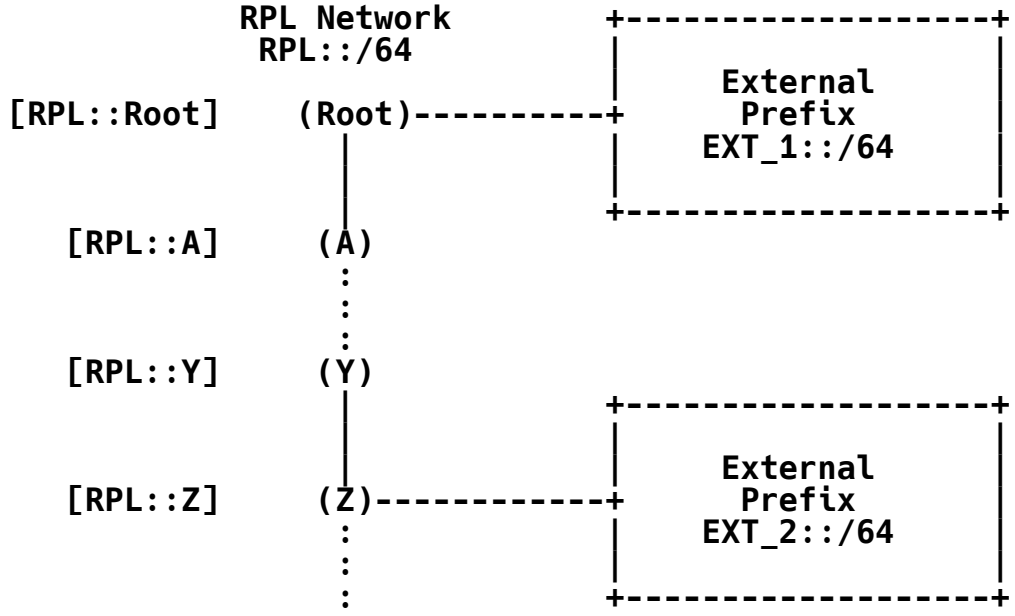


Figure 36: Simple Network Example

In this example, the DODAG root makes a prefix available to the RPL subnet for address autoconfiguration. Here, the entire RPL subnet uses that same prefix, RPL::/64, for address autoconfiguration, though in other implementations more complex/hybrid schemes could be employed.

The DODAG root has connectivity to an external (with respect to that RPL network) prefix EXT_1::/64. The DODAG root may have learned of connectivity to this prefix, for example, via explicit configuration or IPv6 ND on a non-RPL interface. The DODAG root is configured to announce information on the connectivity to this prefix.

Similarly, Node Z has connectivity to an external prefix EXT_2::/64. Node Z also has a sub-DODAG underneath of it.

1. The DODAG root adds a RIO to its DIO messages. The RIO contains the external prefix EXT_1::/64. This information may be repeated in the DIO messages emitted by the other nodes within the DODAG. Thus, the reachability to the prefix EXT_1::/64 is disseminated down the DODAG.

2. Node Z may advertise reachability to the Target network EXT_2::/64 by sending DAO messages using EXT_2::/64 as a Target in the Target option and itself (Node Z) as a parent in the Transit Information option. (In Storing mode, that Transit Information option does not need to contain the address of Node Z). A non-storing root then becomes aware of the 1-hop link (Node Z -- EXT_2::/64) for use in constructing source routes. Node Z may additionally advertise its reachability to EXT_2::/64 to nodes in its sub-DODAG by sending DIO messages with a PIO, with the 'A' flag cleared.

Authors' Addresses

Tim Winter (editor)

EMail: wintert@acm.org

Pascal Thubert (editor)
Cisco Systems
Village d'Entreprises Green Side
400, Avenue de Roumanille
Batiment T3
Biot - Sophia Antipolis 06410
France

Phone: +33 497 23 26 34
EMail: pthubert@cisco.com

Anders Brandt
Sigma Designs
Emdrupvej 26A, 1.
Copenhagen DK-2100
Denmark

EMail: abr@sdesigns.dk

Jonathan W. Hui
Arch Rock Corporation
501 2nd St., Suite 410
San Francisco, CA 94107
USA

EMail: jhui@archrock.com

Richard Kelsey
Ember Corporation
Boston, MA
USA

Phone: +1 617 951 1225
EMail: kelsey@ember.com

Philip Levis
Stanford University
358 Gates Hall, Stanford University
Stanford, CA 94305-9030
USA

EMail: pal@cs.stanford.edu

Kris Pister
Dust Networks
30695 Huntwood Ave.
Hayward, CA 94544
USA

EMail: kpister@dustnetworks.com

Rene Struik
Struik Security Consultancy

EMail: rstruik.ext@gmail.com

JP. Vasseur
Cisco Systems
11, Rue Camille Desmoulins
Issy Les Moulineaux 92782
France

EMail: jpv@cisco.com

Roger K. Alexander
Cooper Power Systems
20201 Century Blvd., Suite 250
Germantown, MD 20874
USA

Phone: +1 240 454 9817
EMail: roger.alexander@cooperindustries.com