Abstract

This document defines a YANG data model for the management of the Address Resolution Protocol (ARP). It extends the basic ARP functionality contained in the ietf-ip YANG data model, defined in RFC 8344, to provide management of optional ARP features and statistics.

The YANG data model in this document conforms to the Network Management Datastore Architecture defined in RFC 8342.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 11, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of
1. Introduction

Basic ARP functionality is supported by the ietf-ip YANG data model, defined in [RFC8344]. This document defines a YANG [RFC7950] data model that extends the basic ARP YANG support to also cover optional ARP features, and ARP related statistics to aid network monitoring and troubleshooting.

This model defines YANG configuration and operational state data nodes both for ARP related functionality formally specified in other RFCs (such as [RFC8344] and [RFC1027]), but also for common ARP behaviour that is often supported on network devices.

Where necessary, the expected behaviour of the YANG data nodes is defined in the YANG model, and this document.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) [RFC8342].
Editorial Note: (To be removed by RFC Editor)

This draft contains several placeholder values that need to be replaced with finalized values at the time of publication. Please apply the following replacements:

- "XXXX" --> the assigned RFC value for this draft both in this draft and in the YANG models under the revision statement.
- The "revision" date in model, in the format XXXX-XX-XX, needs to be updated with the date the draft gets approved. The date also needs to get reflected on the line with <CODE BEGINS>.

1.1. 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 [BCP 14] [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in [RFC8342] and are not redefined here:

- client
- server
- configuration data
- system state
- state data
- intended configuration
- running configuration datastore
- operational state datastore

The following terms are defined in [RFC7950] and are not redefined here:

- augment
- data model
- data node
The terminology for describing YANG data models is found in [RFC7950].

1.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340]

2. Problem Statement

Neither ARP [RFC0826], nor Proxy-ARP [RFC1027], define standard network management configuration models. Instead, network equipment vendors have implemented their own bespoke configuration interfaces and models.

Network operators benefit from having common network management models defined that can be implemented by multiple network equipment manufacturers. This simplifies the operation and management of network devices.

Some, but not all, required ARP functionality has been defined in ietf-ip.yang ([RFC8344]). Providing a standard YANG model that models these optional ARP features, that are fairly widely implemented by network equipment manufacturers, and used by network operators, is beneficial to the general goal of interoperability in the networking industry.

3. Design of the Data Model

This data model intends to describe the processing that a protocol finds the hardware address, also known as Media Access Control (MAC) address, of a host from its known IP address. These tasks include, but are not limited to, configuring dynamic ARP learning, proxy ARP, gratuitous ARP. There are two kind of ARP configurations: global ARP configuration, which is across all interfaces on the device, and per interface ARP configuration.

3.1. ARP Dynamic Learning

As defined in [RFC0826], ARP caching is the method of storing network addresses and the associated data-link addresses in memory for a period of time as the addresses are learned. This minimizes the use of valuable network resources to broadcast for the same address each time a datagram is sent.

There are static ARP cache entries and dynamic ARP cache entries. Static entries, are manually configured and kept in the cache table on a permanent basis which are defined in the ipv4 neighbor list for
each interface in [RFC8344]. Dynamic entries are added by vendor software, kept for a period of time, and then removed. We can specify how long an entry remains in the ARP cache. If we specify a timeout of 0 seconds, entries are never cleared from the ARP cache.

3.2. Proxy ARP

Proxy ARP, defined in [RFC1027], allows a router to respond to ARP requests on behalf of another machine that is not on the same local subnet, offering its own Ethernet media access control (MAC) address. By replying in such a way, the router then takes responsibility for routing packets to the intended destination.

In the case of certain data center network virtualization, as specified in [RFC8014], the proxy ARP can be extended to intercept all ARP requests, including source and target IP addresses in different subnets, and those ARP requests in the same subnet to suppress ARP handling.

3.3. Gratuitous ARP

Gratuitous ARP enables a device to send an ARP Request packet using its own IP address as the destination address. Gratuitous ARP provides the following functions:

- Checks duplicate IP addresses: [RFC5227] uses gratuitous ARP to help detect IP conflicts. When a device receives an ARP request containing a source IP that matches its own, then it knows there is an IP conflict.
- Advertises a new MAC address: Also in RFC 5227, if the MAC address of a host changes because its network adapter is replaced, the host sends a gratuitous ARP packet to notify all hosts of the change before the ARP entry is aged out.
- Notifies an active/standby switchover in a [RFC5798] VRRP backup group: After an active/standby switchover, the master router sends a gratuitous ARP packet in the VRRP backup group to notify the switchover.

3.4. ARP Data Model

This document defines the YANG module "ietf-arp", which has the following structure:
module: ietf-arp
  +--rw arp
    +--rw dynamic-learning? boolean

augment /if:interfaces/if:interface/ip:ipv4:
  +--rw expiry-time? uint32
  +--rw dynamic-learning? boolean
  +--rw proxy-arp
    |  +--rw mode? enumeration
  +--rw gratuitous-arp
    |  +--rw enable? boolean
    |  +--rw interval? uint32
  +--ro statistics
    +--ro discontinuity-time? yang:date-and-time
    +--ro in-requests-pkts? yang:counter32
    +--ro in-replies-pkts? yang:counter32
    +--ro in-gratuitous-pkts? yang:counter32
    +--ro out-requests-pkts? yang:counter32
    +--ro out-replies-pkts? yang:counter32
    +--ro out-gratuitous-pkts? yang:counter32

augment /if:interfaces/if:interface/ip:ipv4/ip:neighbor:
  +--ro remaining-expiry-time? uint32

4. ARP YANG Module

This section presents the ARP YANG module defined in this document.

This module imports definitions from Common YANG Data Types [RFC6991], A YANG Data Model for Interface Management [RFC8343], and A YANG Data Model for IP Management [RFC8344].
prefix ip;
reference "RFC 8344: A Yang Data Model for IP Management";
}

organization
"IETF Routing Area Working Group (rtgwg)";

contact
"WG Web: <http://tools.ietf.org/wg/rtgwg/>
WG List: <mailto: rtgwg@ietf.org>
Editor: Feng Zheng
habby.zheng@huawei.com
Editor: Bo Wu
lana.wubo@huawei.com
Editor: Robert Wilton
rwilton@cisco.com
Editor: Xiaojian Ding
wjswsl@163.com";

description
"Address Resolution Protocol (ARP) management, which includes static ARP configuration, dynamic ARP learning, ARP entry query, and packet statistics collection.

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices."

revision 2019-02-21 {

description
"Init revision";
reference "RFC XXXX: A Yang Data Model for ARP";
}

container arp {

description
"Address Resolution Protocol (ARP)";

leaf dynamic-learning {

type boolean;

default "true";

description
}
"Controls the default ARP learning behavior on all interfaces on the device, unless explicit overridden by the per-interface dynamic-learning leaf:
  true - dynamic learning is enabled on all interfaces by default,
  false - dynamic learning is disabled on all interfaces by default
reference "RFC826: An Ethernet Address Resolution Protocol";
}
}
augment "/if:interfaces/if:interface/ip:ipv4" { description "Augment interfaces with ARP configuration and state.";
container arp {
  description "Address Resolution Protocol (ARP) related configuration and state";
  leaf expiry-time {
    type uint32 {
      range "30..86400";
    }
    units "seconds";
    description "Aging time of a received dynamic ARP entry before it is removed from the cache.";
  }
  leaf dynamic-learning {
    type boolean;
    description "Controls whether dynamic ARP learning is enabled on the interface. If not configured, it defaults to the behavior specified in the per-device /arp/dynamic-learning leaf.
    true - dynamic learning is enabled
    false - dynamic learning is disabled";
  }
  container proxy-arp {
    description "Configuration parameters for proxy ARP";
    leaf mode {
      type enumeration {
        enum disabled {
          description "The system only responds to ARP requests that specify a target address configured on the local interface.";
        }
        enum remote-only {
          description "The system only responds to ARP requests that specify a target address configured on the local interface.";
        }
      }
    }
  }
}
description "The system responds to ARP requests only when the sender and target IP addresses are in different subnets."
}
enum all {
  description "The system responds to ARP requests where the sender and target IP addresses are in different subnets, as well as those where they are in the same subnet.";
}
default "disabled";

description "When set to a value other than 'disable', the local system should respond to ARP requests that are for target addresses other than those that are configured on the local subinterface using its own MAC address as the target hardware address. If the 'remote-only' value is specified, replies are only sent when the target address falls outside the locally configured subnets on the interface, whereas with the 'all' value, all requests, regardless of their target address are replied to.";
reference "RFC1027: Using ARP to Implement Transparent Subnet Gateways";
}

container gratuitous-arp {
  description "Configure gratuitous ARP.";
  reference "RFC5227: IPv4 Address Conflict Detection";
  leaf enable {
    type boolean;
    description "Enable or disable sending gratuitous ARP packet on the interface. The default behaviour is device specific";
  }
  leaf interval {
    type uint32 {
      range "1..86400";
    }
    units "seconds";
    description "The interval, in seconds, between sending gratuitous ARP packet on the interface.";
  }
}

container statistics {

config false;
description
  "ARP per-interface packet statistics

  For all ARP counters, discontinuities in the value can
  occur at re-initialization of the management system and at
  other times as indicated by the value of
  'discontinuity-time'."

leaf discontinuity-time {
  type yang:date-and-time;
description
    "The time on the most recent occasion at which any one or
    more of this interface’s ARP counters suffered a
    discontinuity. If no such discontinuities have occurred
    since the last re-initialization of the local management
    subsystem, then this node contains the time the local
    management subsystem re-initialized itself.";
}

leaf in-requests-pkts {
  type yang:counter32;
description
    "The number of ARP request packets received on this
    interface.";
}

leaf in-replies-pkts {
  type yang:counter32;
description
    "The number of ARP reply packets received on this
    interface.";
}

leaf in-gratuitous-pkts {
  type yang:counter32;
description
    "The number of gratuitous ARP packets received on this
    interface.";
}

leaf out-requests-pkts {
  type yang:counter32;
description
    "The number of ARP request packets sent on this interface.";
}

leaf out-replies-pkts {
  type yang:counter32;
description
    "The number of ARP reply packets sent on this interface.";
leaf out-gratuitous-pkts {
  type yang:counter32;
  description
     "The number of gratuitous ARP packets sent on this
     interface."
}

augment "/if:interfaces/if:interface/ip:ipv4/ip:neighbor" {
  description
     "Augment IPv4 neighbor list with ARP expiry time."
  leaf remaining-expiry-time {
    type uint32;
    units "seconds";
    config false;
    description
      "The number of seconds until the dynamic ARP entry expires and is
      removed from the ARP cache"
  }
}

5. Data Model Examples

This section presents a simple example of configuring static ARP entries and dynamic learning, based on the YANG modules specified in Section 4.

5.1. Static ARP Entries

Requirement: Enable static ARP entry configuration on interface (defined in [RFC8344]).

```xml
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
      <neighbor>
        <ip-address>192.0.2.1</ip-address>
        <link-layer-address>00e0:fc01-0000</link-layer-address>
        <origin>static</origin>
      </neighbor>
    </ipv4>
  </interfaces>
</config>
```
5.2. ARP Dynamic Learning

Requirement: Disable ARP dynamic learning configuration.

```xml
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <arp xmlns="urn:ietf:params:xml:ns:yang:ietf-arp">
    <dynamic-learning>false</dynamic-learning>
  </arp>
</config>
```

6. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

Name: ietf-arp
Namespace: urn:ietf:params:xml:ns:yang:
Prefix: arp
Reference: RFC XXXX

7. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest
RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- **arp/dynamic-learning**: This leaf is used to enable ARP dynamic learning on all interfaces. ARP dynamic learning could allow an attacker to inject spoofed traffic into the network, e.g. denial-of-service attack.

- **interface/ipv4/arp/proxy**: These leaves are used to enable ARP proxy on interface. They could allow traffic to be mis-configured (denial-of-service attack).

- **interface/ipv4/arp/gratuitous-arp**: This leaf is used to enable sending gratuitous ARP packet on an interface. This configuration could allow an attacker to inject spoofed traffic into the network, e.g. man-in-the-middle attack.

8. Acknowledgments

The authors wish to thank Alex Campbell and Reshad Rahman, Qin Wu, Tom Petch, many others for their helpful comments.

9. References

9.1. Normative References


9.2. Informative References


Authors’ Addresses

Feng Zheng
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: habby.zheng@huawei.com
