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 February 25, 2019.

Copyright Notice

Copyright (c) 2018 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 Regarding IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review those documents prior to using this document.
1. Introduction

This document defines a YANG [RFC7950] data model for the Address Resolution Protocol [RFC0826] implementation and identification of some common properties within a device. Devices have common properties that need to be configured and monitored in a standard way. This document is intended to present universal ARP protocol configuration and many vendors can implement it.

The data model converts configuration of system parameters of ARP, such as static ARP entries, timeout for dynamic ARP entries, interface ARP, proxy ARP, and so on. It also provides information about running state of ARP implementations.

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 many 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.

- Revision date in model, in the format 2018-08-01 needs to get 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

This document defines a YANG [RFC7950] configuration data model that may be used to configure the ARP feature running on a system. Data model "ietf-ip" [RFC8344] covers the address mapping functionality. However, this functionality is strictly dependent on IPv4 networks, and many ARP related functionalities are missing, e.g. device global ARP entries and control, configuration related to dynamic ARP learning, proxy ARP, gratuitous ARP, etc.

The data model makes use of the YANG "feature" construct which allows implementations to support only those ARP features that lie within their capabilities. It is intended this model be used by service providers who manipulate devices from different vendors in a standard way.

This model can be used to configure the ARP applications for discovering the link layer address associated with a given Internet layer address.

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, adding a static entry in the ARP cache, 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

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. 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 [RFC1027] can be configured to enable the switch to respond to ARP queries for network addresses by offering its own Ethernet media access control (MAC) address. With proxy ARP enabled, the switch captures and routes traffic to the intended destination.

3.3. gratuitous ARP

Gratuitous ARP requests help detect duplicate IP addresses. A gratuitous ARP is a broadcast request for a router’s own IP address. If a router or switch sends an ARP request for its own IP address and no ARP replies are received, the router- or switch-assigned IP address is not being used by other nodes. However, if a router or switch sends an ARP request for its own IP address and an ARP reply is received, the router- or switch-assigned IP address is already being used by another node.

3.4. ietf-arp Module

This module has one top level container, ARP, which consists of two second level containers, which are used for static entries configuration and global parameters control.
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].

<CODE BEGINS>file "ietf-arp@2018-08-01.yang"
module ietf-arp {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-arp";
  prefix arp;

  import ietf-inet-types {

  }

  module: ietf-arp
    +--rw arp
      +--rw dynamic-learning? boolean
      +--rw proxy-arp? boolean
      +--rw global-static-entries {global-static-entries}?
        +--rw static-entry* [ip-address]
          +--rw ip-address inet:ipv4-address-no-zone
          +--rw mac-address yang:mac-address

    augment /if:interfaces/if:interface:
      +--rw arp
        +--rw expiry-time? uint32
        +--rw learn-disable? boolean
        +--rw proxy
          | +--rw mode? enumeration
        +--rw probe
          | +--rw interval? uint8
          | +--rw times? uint8
          | +--rw unicast? boolean
        +--rw gratuitous
          | +--rw enable? boolean
          | +--rw interval? uint32
          | +--rw drop? boolean
        +--ro statistics
          +--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

prefix inet;
  reference "RFC 6991: Common YANG Data Types";
}

import ietf-yang-types {
  prefix yang;
  reference "RFC 6991: Common YANG Data Types";
}

import ietf-interfaces {
  prefix if;
  reference "RFC 8343: A Yang Data Model for Interface Management";
}

import ietf-ip {
  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: Xiaojian Ding
    wjsws1@163.com
  Editor: Feng Zheng
    habby.zheng@huawei.com
  Editor: Robert Wilton
    rwilton@cisco.com";

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

  Copyright (c) 2016 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 2018-08-01 {
  description
feature global-static-entries {
  description "This feature indicates that the device allows static entries to be configured globally.";
}

container arp {
  description "Address Resolution Protocol (ARP) management, which includes static ARP configuration, dynamic ARP learning, ARP entry query, and packet statistics collection.";
  leaf dynamic-learning {
    type boolean;
    default "true";
    description "Controls the default dynamic ARP learning behavior on all interfaces on the device:
      true - dynamic learning is enabled on all interfaces by default,
      false - dynamic learning is disabled on all interfaces by default";
  }
  leaf proxy-arp {
    type boolean;
    default "true";
    description "Controls the default proxy ARP behavior on all interfaces on the device:
      true - proxy ARP is enabled on interfaces by default,
      false - proxy ARP is disabled on interfaces by default";
  }
  container global-static-entries {
    if-feature "global-static-entries";
    description "Set a global static ARP entry, which is independent of the interface.";
    list static-entry {
      key "ip-address";
      description "List of ARP static entries that can be configured globally.";
      leaf ip-address {
        type inet:ipv4-address-no-zone;
        description "";
      }
    }
  }
}
"IP address, in dotted decimal notation."
}
leaf mac-address {
    type yang:mac-address;
    mandatory true;
    description
    "MAC address in the format of H-H-H, in which H is a hexadecimal number of 1 to 4 bits."
}
}
}
augment "/if:interfaces/if:interface" {
    description
    "Augment interfaces with ARP configuration and state.";
    container arp {
        description
        "Dynamic ARP related configuration and state";
        leaf expiry-time {
            type uint32 {
                range "60..86400";
            }
            units "seconds";
            description
            "Aging time of a dynamic ARP entry."
        }
        leaf learn-disable {
            type boolean;
            default "false";
            description
            "Whether dynamic ARP learning is disabled on an interface: 
            If the value is True, dynamic ARP learning is disabled. 
            If the value is False, dynamic ARP learning is enabled.";
        }
        container proxy {
            description
            "Configuration parameters for proxy ARP";
            leaf mode {
                type enumeration {
                    enum DISABLE {
                        description
                        "The system should not respond to ARP requests that do not specify an IP address configured on the local subinterface as the target address.";
                    }
                    enum REMOTE_ONLY {
                        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 "DISABLE";
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 probe {
  description
    "Common configuration parameters for all ARP probe.";
  leaf interval {
    type uint8 {
      range "1..5";
    }
    units "second";
    description
      "Interval for detecting dynamic ARP entries.";
  }
  leaf times {
    type uint8 {
      range "0..10";
    }
    description
      "Number of aging probe attempts for a dynamic ARP entry. If a device does not receive an ARP reply message after the number of aging probe attempts reaches a specified number, the dynamic ARP entry is deleted.";
  }
  leaf unicast {

type boolean;
default "false";
description
"Send unicast ARP aging probe messages for a dynamic ARP
entry.";
}
}
container gratuitous-arp {
    description
    "Configure gratuitous ARP.";
    leaf enable {
        type boolean;
        default "false";
        description
        "Enable or disable sending gratuitous ARP packet on
        interface.";
    }
    leaf interval {
        type uint32 {
            range "1..86400";
        }
        units "second";
        description
        "The interval of sending gratuitous ARP packet on the
        interface.";
    }
    leaf drop {
        type boolean;
        default "false";
        description
        "Drop the receipt of gratuitous ARP packets on the
        interface.";
    }
}
}
container statistics {  
    config false;
    description
    "IP ARP Statistics information on interfaces";
    leaf in-requests-pkts {
        type yang:counter32;
        description
        "Total ARP requests received";
    }
    leaf in-replies-pkts {
        type yang:counter32;
        description
        "Total ARP replies received";
    }
}
leaf in-gratuitous-pkts {
    type yang:counter32;
    description "Total gratuitous ARP received";
}

leaf out-requests-pkts {
    type yang:counter32;
    description "Total ARP requests sent";
}

leaf out-replies-pkts {
    type yang:counter32;
    description "Total ARP replies sent";
}

leaf out-gratuitous-pkts {
    type yang:counter32;
    description "Total gratuitous ARP sent";
}

augment "/if:interfaces/if:interface/ip:ipv4/ip:neighbor" {
    description "Augment neighbor list with parameters of ARP, eg., support for remaining expiry time query on interfaces.";
    leaf remaining-expiry-time {
        type uint32;
        config false;
        description "Remaining expiry time of a dynamic ARP entry. ";
    }
}

5. Data Model Examples

This section presents a simple but complete 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 global configuration (not rely on interface).

```xml
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <arp xmlns="urn:ietf:params:xml:ns:yang:ietf-arp">
    <static-tables>
      <ip-address> 10.2.2.3 </ip-address>
      <mac-address> 00e0-fc01-0000 </mac-address>
    </static-tables>
  </arp>
</config>
```

Requirement:
Enable static ARP entry configuration on interface (defined in draft [I-D.ietf-netmod-rfc7277bis]).

```xml
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
    <neighbor>
      <ip-address> 10.2.2.3 </ip-address>
      <mac-address> 00e0-fc01-0000 </mac-address>
      <if-name> GE1/0/1 </if-name>
    </neighbor>
  </ipv4>
</config>
```

5.2. ARP Dynamic Learning

Requirement:
Enable 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">
    <if-name> GE1/0/1 </if-name>
    <expire-time>1200</expire-time>
    <learn-disable>false</learn-disable>
    <proxy-enable>false</proxy-enable>
    <probe>
      <interval>5</interval>
      <times>3</times>
    </probe>
    <unicast>false</unicast>
  </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 [RFC7950].

Name: ietf-arp
Prefix: arp
Reference: RFC XXXX

7. Security Considerations

The YANG module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available 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.

arp/proxy-arp and arp/proxy: These leaves are used to enable ARP proxy on interface. They could allow traffic to be mis-configured (denial-of-service attack).
arp/global-static-entries/static-entry: This list specifies ARP static entries configured on the device. By modifying this information, an attacker can cause a node to either ignore messages destined to it or accept messages it would otherwise ignore.

/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