

MBONED Working Group
Internet Draft
Intended status: Standards Track
Expires: April 09, 2026

Y. Liu
China Mobile
C. Lin
New H3C Technologies
Z. Zhang
ZTE
X. Geng
Huawei
V. Kumar Nagaraj
Juniper Networks
October 9, 2025

A YANG Data Model for Automatic Multicast Tunneling (AMT)
[draft-ietf-mboned-amt-yang-05](#)

Abstract

This document defines ~~a YANG data model for the configuration and management of Automatic Multicast Tunneling (AMT) protocol operations.~~

Commenté [MB1]: This is covered by management.

As a reminder C is one of the FCAPS management functions.

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 April 09, 2026.

Copyright Notice

Copyright (c) 2025 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 publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction..... | 2 |
| 1.1. Terminology..... | 3 |
| 1.2. Conventions Used in This Document..... | 3 |
| 1.3. Tree Diagrams..... | 3 |
| 2. Model Overview..... | 3 |
| 2.1. Prefixes in Data Node Names..... | 4 |
| 3. AMT YANG Module..... | 4 |
| 3.1. Tree View..... | 4 |
| 3.2. Yang Module..... | 7 |
| 4. Data Model Example..... | 21 |
| 5. Security Considerations..... | 21 |
| 6. IANA Considerations..... | 23 |
| 6.1. IETF XML Registry..... | 23 |
| 6.2. YANG Module Names Registry..... | 23 |
| 7. References..... | 24 |
| 7.1. Normative References..... | 24 |
| 7.2. Informative References..... | 25 |
| Authors' Addresses..... | 26 |

1. Introduction

[RFC7450] introduces the protocol definition of the Automatic Multicast Tunneling (AMT) for delivering multicast traffic from sources in a multicast-enabled network to receivers that lack multicast connectivity to the source network. The protocolAMT uses UDP encapsulation and unicast replication to provide this functionality.

[RFC8777] updates [RFC7450] by modifying the relay discovery process. It defines DNS Reverse IP AMT Discovery (DRIAD) mechanism for AMT gateways to discover AMT relays that are capable of forwarding multicast traffic from a known source IP address.

This document defines a YANG data models for configuring and managing AMT ProtocolProtocol.

~~The YANG module defined in this document conforms to the Network Management Datastore Architecture (NMDA) [RFC8342].~~

Commenté [MB2]: The new guidance in RFC8704bis is to call out when this is not compliant.

1.1.2. Terminology & Notation Conventions

2.1. Conventions Used in This Document

~~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.~~

2.2 Terminology

The terminology for describing YANG data models is found in [RFC6020] and [RFC7950], including:

- o augment

- o data model
- o data node
- o identity
- o module

~~The following abbreviations are used in this document and the defined model:~~

~~AMT: Automatic Multicast Tunneling [RFC7450].~~

~~XX~~

1.2. Conventions Used in This Document

~~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.~~

1.2.3. Tree Diagrams

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

2. Data Model Overview

~~AMT YANG data models are defined in this document.~~

~~The `ietf-amt.yang` data model. The AMT data model provides the methods for configuring and managing AMT protocol. It includes:~~

- o Parameters of ~~AMT relay service~~, such as Relay Discovery Address ([Section 4.1.5 of \[RFC7450\]](#)), Relay Address ([Section 4.1.5 of \[RFC7450\]](#)), service switch, the maximum number of tunnels, and secret key timeout.

- o Parameters of ~~AMT gateway service~~, such as Relay Discovery Address, Relay Address, Discovery Timeout, Request Timeout and Maximum Retransmission Count.

- o AMT tunnel information, such as endpoint IP address and UDP port number, local IP address and UDP port number.

- o DNS ~~Resource record (RR)~~ used by an AMT relay service.

2.1. Prefixes in Data Node Names

~~In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the~~

Commenté [MB3]: This is not needed as this is about one entry. This is already expanded in the intro.

Commenté [MB4]: Please add key AMT terms that are used in the doc: AMT relay/GW/etc.

Commenté [MB5]: I suggest to add a figure to show the various components. This would help readers understand which parts are covered by the module and also the interactions between various components.

Commenté [MB6]: As this is about a YANG file.

Please check the discussion about data model vs module in Section 2.5 of RFC8407bis

Commenté [MB7]: Should be defined in the terminology section

Commenté [MB8]: Add where this parameter is defined.

Commenté [MB9]: Add where this is defined.

Commenté [MB10]: Should be defined in the terminology section

Commenté [MB11]: Consider adding pointers where each of these parameters is defined.

Commenté [MB12]: There is no actions in the doc

corresponding YANG module, as shown in Table 1 summarizes the prefixes used in this document.

| Prefix | YANG module | Reference |
|----------|--------------------|-----------|
| inet | ietf-inet-types | [RFC6991] |
| rt-types | ietf-routing-types | [RFC8294] |
| rt | ietf-routing | [RFC8349] |
| yang | ietf-yang-types | [RFC6991] |
| if | ietf-interfaces | [RFC8343] |

Table 1: Prefixes and Corresponding YANG Modules

Commenté [MB13]: Please update as this is obsoleted by RFC9911.

Commenté [MB14]: Move this to be under the YANG module as prefixes are used there.

3. AMT YANG Module

3.1. Tree View

The complete tree of the "ietf-amt.yang" data model YANG module is represented as
Following in Figure X. See [RFC8340] for an explanation of the symbols used.

The AMT YANG is module model augments the core routing data model YANG module "ietf-routing"
specified in [RFC8349]. Specifically, The the AMT model YANG module augments
"/rt:routing/rt:control-plane-protocols".

Commenté [MB15]: Already mentioned in the terminology section

```
module: ietf-amt
  augment /rt:routing/rt:control-plane-protocols:
    +--rw amt!
      +--rw relay
        | +--rw relay-addresses
        | | +--rw relay-address* [family]
        | | | +--rw family          identityref
        | | | +--rw anycast-prefix  inet:ip-prefix
        | | | +--rw local-address   inet:ip-address
        | +--rw tunnel-limit?
        | +--rw secret-key-timeout? uint32
      +--ro tunnels
        | +--ro tunnel* [gateway-address gateway-port]
        | | +--ro gateway-address  inet:ip-address
        | | +--ro gateway-port     inet:port-number
        | | +--ro local-address    inet:ip-address
        | | +--ro local-port       inet:port-number
        | | +--ro state            enumeration
        | | +--ro multicast-flows
          | | | +--ro multicast-flow* [source-address
          | | | | | group-address]
          | | | +--ro source-address ip-multicast-source-address
          | | | +--ro group-address
```

Commenté [MB16]: Relay and GW may not supported by the same device. No?

I would define two features: relay and gw to conditionally trim part of the model that applies to each components

Commenté [MB17]: Do we need to say something about private key provisioning?

```

| | | | rt-types:ip-multicast-group-address
| | | +--ro multicast-group-num yang:gauge32
| | | +--ro request-message-count
| | | | yang:zero-based-counter64
| | | +--ro membership-query-message-count
| | | | yang:zero-based-counter64
| | | +--ro membership-update-message-count
| | | | yang:zero-based-counter64
| | +--rw relay-dns-resource-records
| | +--rw relay-dns-resource-record* [source-address]
| | | +--rw source-address inet:ip-address
| | | +--rw precedence? uint32
| | | +--rw d-flag? boolean
| | | +--rw relay-type? enumeration
| | | +--rw discovery-address? inet:ip-address
| | | +--rw domain-name? inet:domain-name
| +--ro relay-message-statistics
| | +--ro received
| | | +--ro relay-discovery yang:zero-based-counter64
| | | +--ro request yang:zero-based-counter64
| | | +--ro membership-update yang:zero-based-counter64
| | | +--ro teardown yang:zero-based-counter64
| | +--ro sent
| | | +--ro relay-advertisement yang:zero-based-counter64
| | | +--ro membership-query yang:zero-based-counter64
| | +--ro error
| | | +--ro incomplete-packet yang:zero-based-counter64
| | | +--ro invalid-mac yang:zero-based-counter64
| | | +--ro unexpected-type yang:zero-based-counter64
| | | +--ro invalid-relay-discovery-address
| | | | yang:zero-based-counter64
| | | +--ro invalid-membership-request-address
| | | | yang:zero-based-counter64
| | | +--ro invalid-membership-update-address
| | | | yang:zero-based-counter64
| | | +--ro incomplete-relay-discovery-messages
| | | | yang:zero-based-counter64
| | | +--ro incomplete-membership-request-messages
| | | | yang:zero-based-counter64
| | | +--ro incomplete-membership-update-messages
| | | | yang:zero-based-counter64
| | | +--ro no-active-gateway yang:zero-based-counter64
| | | +--ro invalid-inner-header-checksum
| | | | yang:zero-based-counter64
| | | +--ro gateways-timed-out yang:gauge64
| +--rw gateway
| | +--rw pseudo-interfaces
| | | +--rw pseudo-interface* [interface]
| | | | +--rw interface if:interface-ref
| | | | +--rw discovery-method enumeration
| | | | +--rw relay-discovery-address? inet:ip-address
| | | | +--rw relay-address? inet:ip-address
| | | | +--rw upstream-interface? if:interface-ref
| | | | +--rw discovery-timeout? uint32
| | | | +--rw discovery-retrans-count? uint32
| | | | +--rw request-timeout? uint32
| | | | +--rw request-retrans-count? uint32
| | | +--rw dest-unreach-retry-count? uint32

```

Commenté [MB18]: I would position this one before tunnels.

Commenté [MB19]: Does this corresponds to the D-bit?
I would use that name, not flag

Commenté [MB20]: Some | are missing

```

|   +-rw relay-port?          inet:port-number
|   +-ro local-address?      inet:ip-address
|   +-ro local-port?         inet:port-number
|   +-ro tunnel-state        enumeration
|   +-ro relay-discovery-message-count
|       |           yang:zero-based-counter64
|   +-ro relay-advertisement-message-count
|       |           yang:zero-based-counter64
|   +-ro request-message-count
|       |           yang:zero-based-counter64
|   +-ro membership-query-message-count
|       |           yang:zero-based-counter64
|   +-ro membership-update-message-count
|       |           yang:zero-based-counter64
+-ro gateway-message-statistics
    +-ro received
    |   +-ro relay-advertisement yang:zero-based-counter64
    |   +-ro membership-query   yang:zero-based-counter64
    +-ro sent
        +-ro relay-discovery     yang:zero-based-counter64
        +-ro request            yang:zero-based-counter64
        +-ro membership-update  yang:zero-based-counter64
        +-ro teardown           yang:zero-based-counter64

```

Figure X: AMT Tree Structure

3.2. Yang-YANG Module Module

This document imports modules defined in [RFC9961], [RFC8294], [RFC8343], and [RFC8349].

```

<CODE BEGINS> file "ietf-amt@2022-02-17.yang"
module ietf-amt {
    yang-version "1.1";
    namespace "urn:ietf:params:xml:ns:yang:ietf-amt";
    prefix amt;

    import ietf-inet-types {
        prefix inet;
        reference
            "RFC 6991: Common YANG Data Types, Section 4";
    }

    import ietf-yang-types {
        prefix yang;
        reference
            "RFC 6991: Common YANG Data Types, Section 3";
    }

    import ietf-routing-types {
        prefix rt-types;
        reference
            "RFC 8294: Common YANG Data Types for the Routing Area";
    }

    import ietf-interfaces {
        prefix if;
    }
}

```

Commenté [MB21]: You may consider moving the full tree to an appendix, but use snippets to explain the role/meaning of main parts of the hierarchy.

You can refer to <https://www.rfc-editor.org/rfc/rfc9834.html#name-the-attachment-circuit-serv> for an example.

```
reference
    "RFC 8343: A YANG Data Model for Interface Management";
}

import ietf-routing {
    prefix rt;
    reference
        "RFC 8349: A YANG Data Model for Routing Management
            (NMDA Version)";
}

organization
    "IETF Multicast Backbone Deployment (MBONED) Working Group";

contact
    "WG Web: <https://datatracker.ietf.org/wg/mboned/>
    WG List: MBONED <mailto:mboned@ietf.org>

    Editor: Yisong Liu
             <mailto:liuyisong@chinamobile.com>
    Editor: Changwang Lin
             <mailto:linchangwang.04414@h3c.com>
    Editor: Zheng(Sandy) Zhang
             <mailto:zhang.zheng@zte.com.cn>
    Editor: Xuesong Geng
             <mailto:gengxuesong@huawei.com>
    Editor: Vinod Kumar Nagaraj
             <mailto:vinkumar@juniper.net>";
```

description
"This module describes a YANG model for ~~configuring and~~
~~managing the AMT pProtocol.~~

~~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 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.";~~

~~This YANG model conforms to the Network Management Datastore Architecture (NMDA) as described in RFC 8342.~~

Copyright (c) ~~2022-2026~~ 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 Revised BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

All revisions of IETF and IANA published modules can be found at the YANG Parameters registry group (<https://www.iana.org/assignments/yang-parameters>).

This version of this YANG module is part of RFC XXXX; see the

RFC itself for full legal notices.

~~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 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here."~~

```
revision 20220202-1102-17-26 {
    description
        "Initial Version";
    reference
        "RFC XXXX: A YANG Data Model for Automatic Multicast Tunneling";
}
```

```
identity address-family {
    description
        "Base identity from which identities describing address
         families are derived.";
}
```

```
identity ipv4 {
    base address-family;
    description
        "This identity represents an IPv4 address family.";
}
```

```
identity ipv6 {
    base address-family;
    description
        "This identity represents an IPv6 address family.";
}
```

```
typedef ip-multicast-source-address {
    type union {
        type rt-types:ipv4-multicast-source-address;
        type rt-types:ipv6-multicast-source-address;
    }
    description
        "This type represents a version-neutral IP multicast source
         address. The format of the textual representation implies
         the IP versionaddress family.";
```

```
augment "/rt:routing/rt:control-plane-protocols" {
    description
        "AMT augmentation to the routing instance model.";
    container amt {
        description
            "Configuration Management parameters for the AMT protocol.";
        container relay {
            description
                "Parameters of AMT relay service.";
            container relay-addresses {
                description
                    "Parameters of AMT relay addresses.";

                list relay-address {
```

Commenté [MB22]: Why not reusing address-family from RFC 8349 that we already import here.

Commenté [MB23]: Please add reference clauses for each of the parameters.

Commenté [MB24]: As this can be used for retrieval as well

Commenté [MB25]: No need to repeat the parent prefix.

```

key "family";
description
    "Each entry contains parameters for an AMT relay
     address identified by the 'family' key. Under normal
operation, these addresses belong to the same address family indicated by
the 'family'. Any mismatch is indication of abnormal configurations.;
leaf family {
    type identityref {
        base address-family;
    }
    mandatory true;
    description
        "Indicates the Address-address family for the entry.;";
}
leaf anycast-prefix {
    type inet:ip-prefix;
    description
        "The anycast IP prefix of AMT relay discovery
         address which is used when sending discovery
         messages to a relay.";
}
leaf local-address {
    type inet:ip-address;
    description
        "The A unicast IP address of the AMT relay address
         which is obtained as a result of the discovery
         process.;";
}
leaf tunnel-limit {
    type uint32;
    description
        "The total number of endpoints.";
}
leaf secret-key-timeout {
    type uint32;
    description
        "The timeout interval of secret key.";
}
container tunnels {
    config false;
    description
        "[The AMT tunnel information on the relay.]";
list tunnel {
    key "gateway-address gateway-port";
    description
        "An entry of AMT tunnelRecords a tunnel entry.;";
    leaf gateway-address {
        type inet:ip-address;
        description
            "The IP address of an AMT gateway.";
    }
    leaf gateway-port {
        type inet:port-number;
        description
            "The UDP port number of an AMT gateway.";
    }
}

```

Commenté [MB26]: Hmm, we do say this is a «key». This is redundant.

Commenté [MB27]: This means that we allow only for one and only one address per address family?

Is that really intended?

Commenté [MB28]: What is the typical usage of a list that has only the family key?

I guess this can be used for a GET to query local configuration. If so, please add such intended use in the description.

Commenté [MB29]: Please consider expanding a bit the description to be useful for readers.

```

leaf local-address {
    type inet:ip-address;
    description
        "The local IP address of the AMT relay.";
}
leaf local-port {
    type inet:port-number;
    description
        "The local UDP port of the AMT relay.";
}
leaf state {
    type enumeration {
        enum up {
            description
                "The AMT tunnel has been successfully
                established.";
        }
        enum establishing {
            description
                "The AMT tunnel is being established.";
        }
    }
    description
        "The state of AMT tunnel.";
}
container multicast-flows {
    config false;
    description
        "The multicast flow information in the AMT tunnel.";
    list multicast-flow {
        key "source-address group-address";
        description
            "An entry of multicast flow. Records the
characteristics of a multicast flow.";
        leaf source-address {
            type ip-multicast-source-address;
            description
                "The source IP address of a multicast flow.";
        }
        leaf group-address {
            type rt-types:ip-multicast-group-address;
            description
                "The group IP address of a multicast flow.";
        }
    }
}
leaf multicast-group-num {
    type yang:gauge32;
    description
        "Number of multicast groups.";
}
leaf request-message-count {
    type yang:zero-based-counter64;
    description
        "Number of AMT request messages received
        in the tunnel.";
}
leaf membership-query-message-count {

```

Commenté [MB30]: Couldn't we simply reuse the udp-client grouping from draft-ietf-netconf-udp-client-server?

Commenté [MB31]: Are we confident these are the one possible state?

Wouldn't there be other state to track failed/discriminated/etc.

I suggest we define those as identities so that we have something that we can extend in the future if needed.

Commenté [MB32]: Please expand a bit the description.

Commenté [MB33]: Idem as above. Please add an Operational consideration that these are supposed to be from the same address family.

Any mismatch indicates an anomaly.

```

        type yang:zero-based-counter64;
        description
            "Number of AMT membership query messages sent
             in the tunnel.";
    }
leaf membership-update-message-count {
    type yang:zero-based-counter64;
    description
        "Number of AMT membership update messages received
         in the tunnel.";
}
container relay-dns-resource-records {
    description
        "The DNS Resource records Records (RRs) of the AMT
relay.";
    list relay-dns-resource-record {
        key "source-address";
        description
            "An entry of AMT relay resource record specifies an RR
entry.";
        leaf source-address {
            type inet:ip-address;
            description
                "The IP address of multicast sender.";
        }
        leaf precedence {
            type uint32;
            description
                "The precedence of this record.";
        }
        leaf d-flag {
            type boolean;
            default false;
            description
                "If the D-bit is set to true, the gateway MAY
                 send an AMT Request message directly to the
                 discovered relay address without first
                 sending an AMT Discovery message.
                 If the D-bit is set to false, the gateway MUST
                 receive an AMT relay advertisement message
                 for an address before sending an AMT
                 Request message to that address.";
        }
    }
    leaf relay-type {
        type enumeration {
            enum empty {
                value 0;
                description
                    "The relay field is empty.";
            }
            enum ipv4-address {
                value 1;
                description
                    "The relay field contains a 4-octet IPv4
                     address.";
            }
        }
    }
}

```

Commenté [MB34]: Should we have a record of discontinuity time?

I think we need one here

Commenté [MB35]: Need more description so that this be useful.

Also, add a reference clause.

Commenté [MB36]: Please add a reference clause

Commenté [MB37]: Can a relay be identified by multiple types?

If so, how this is supposed to work with this structure?

If not, please say so in the description.

```

enum ipv6-address {
    value 2;
    description
        "The relay field contains a 16-octet IPv6
        address.";
}
enum domain-name {
    value 3;
    description
        "The relay field contains a wire-encoded
        domain name.";
}
description
    "The type of Relay address.";
}

leaf discovery-address {
    type inet:ip-address;
    description
        "The IP address of AMT relay discovery address.";
}
leaf domain-name {
    type inet:domain-name;
    description
        "The wire-encoded domain name of AMT relay.";
}
}

container relay-message-statistics {
    config false;
    description
        "Message statistics of an AMT relay.";
    container received {
        description
            "Received message statistics of AMT relay.";
        leaf relay-discovery {
            type yang:zero-based-counter64;
            description
                "Number of AMT relay discovery messages
                received.";
        }
        leaf request {
            type yang:zero-based-counter64;
            description
                "Number of AMT membership request messages
                received.";
        }
        leaf membership-update {
            type yang:zero-based-counter64;
            description
                "Number of AMT membership update messages
                received.";
        }
        leaf teardown {
            type yang:zero-based-counter64;
            description
                "Number of AMT teardown messages received.";
        }
    }
}

```

Commenté [MB38]: Please indicate how this is supposed to be used.

Commenté [MB39]: Do these depend on the value of relay-type?

Commenté [MB40]: Idem as above, should a discontinuity time be provided?

The comment applies for similar parts of the module below.

```

}
container sent {
    description
        "Sent message statistics of AMT relay.";
    leaf relay-advertisement {
        type yang:zero-based-counter64;
        description
            "Number of AMT relay advertisement messages sent.";
    }
    leaf membership-query {
        type yang:zero-based-counter64;
        description
            "Number of AMT membership query messages sent.";
    }
}
container error {
    description
        "Error message statistics of AMT relay.";
    leaf incomplete-packet {
        type yang:zero-based-counter64;
        description
            "Number of messages received with length errors
             so severe that further classification could not
             occur.";
    }
    leaf invalid-mac {
        type yang:zero-based-counter64;
        description
            "Number of messages received with an invalid
             message authentication code (MAC).";
    }
    leaf unexpected-type {
        type yang:zero-based-counter64;
        description
            "Number of messages received with an unknown
             message type specified.";
    }
    leaf invalid-relay-discovery-address {
        type yang:zero-based-counter64;
        description
            "Number of AMT relay discovery messages
             received with an address other than the
             configured anycast address.";
    }
    leaf invalid-membership-request-address {
        type yang:zero-based-counter64;
        description
            "Number of AMT membership request messages
             received with an address other than the
             configured AMT local address.";
    }
    leaf invalid-membership-update-address {
        type yang:zero-based-counter64;
        description
            "Number of AMT membership update messages
             received with an address other than the
             configured AMT local address.";
    }
}

```

```

leaf incomplete-relay-discovery-messages {
    type yang:zero-based-counter64;
    description
        "Number of AMT relay discovery messages
         received that are not fully formed.";
}
leaf incomplete-membership-request-messages {
    type yang:zero-based-counter64;
    description
        "Number of AMT membership request messages
         received that are not fully formed.";
}
leaf incomplete-membership-update-messages {
    type yang:zero-based-counter64;
    description
        "Number of AMT membership update messages
         received that are not fully formed.";
}
leaf no-active-gateway {
    type yang:zero-based-counter64;
    description
        "Number of AMT membership update messages
         received for a tunnel that does not exist
         for the gateway that sent the message.";
}
leaf invalid-inner-header-checksum {
    type yang:zero-based-counter64;
    description
        "Number of AMT membership update messages
         received with an invalid IP checksum.";
}
leaf gateways-timed-out {
    type yang:gauge64;
    description
        "Number of gateways that timed out because
         of inactivity.";
}
}
}
}
// relay
container gateway {
    description
        "Parameters of AMT gateway service.";
    container pseudo-interfaces {
        description
            "Parameters of AMT pseudo-interface.";
        list pseudo-interface {
            key "interface";
            description
                "An entry of AMT pseudo-interface.";
            leaf interface {
                type if:interface-ref;
                description
                    "Indicates a Pseudo-pseudo interface.";
            }
            leaf discovery-method [] {
                type enumeration {
                    enum by-amt-solicit {

```

Commenté [MB41]: Add to the terminolgy section.

Commenté [MB42]: Do we expect that other methods to be defined in the future?

If so, please define this as identities.

```
        description
            "Find the relay address by sending an AMT
             Discovery message.";
    }
    enum by-dns-reverse-ip {
        description
            "Find the relay address by DNS reverse IP
             AMT Discovery.";
    }
}
description
    "The method used to discover the relay address.";
}
leaf relay-discovery-address {
    type inet:ip-address;
    description
        "IP address of the AMT relay discovery address.";
}
leaf relay-address {
    type inet:ip-address;
    description
        "IP address of the AMT relay address.";
}
leaf upstream-interface {
    type if:interface-ref;
    description
        "Upstream interface.";
}
leaf discovery-timeout {
    type uint32;
    description
        "Initial time to wait for a response to
         a Relay Discovery message.";
}
leaf discovery-retrans-count {
    type uint32;
    description
        "Maximum number of Relay Discovery retransmissions
         to allow before terminating relay discovery
         and reporting an error.";
}
leaf request-timeout {
    type uint32;
    description
        "Initial time to wait for a response
         to a Request message";
}
leaf request-retrans-count {
    type uint32;
    description
        "Maximum number of Request retransmissions
         to allow before abandoning a relay and restarting
         relay discovery or reporting an error.";
}
leaf dest-unreach-retry-count {
    type uint32;
    description
        "The maximum number of times a gateway should
```

```

        attempt to send the same Request or Membership
        Update message after receiving an ICMP Destination
        Unreachable message.";
    }
leaf relay-port {
    type inet:port-number;
    description
        "The UDP port number of AMT relay.";
}
leaf local-address {
    type inet:ip-address;
    config false;
    description
        "The local IP address of this AMT tunnel.";
}
leaf local-port {
    type inet:port-number;
    config false;
    description
        "The local UDP port number of this AMT tunnel.";
}
leaf tunnel-state {
    type enumeration {
        enum initial {
            description
                "Initial state.";
        }
        enum discovering {
            description
                "The Relay Discovery message has been sent
                and is waiting for the Advertisement message.";
        }
        enum requesting {
            description
                "The Request message has been sent,
                waiting for the Query message.";
        }
        enum up {
            description
                "The AMT tunnel is Established.";
        }
    }
    config false;
    description
        "The tunnel's state.";
}
leaf relay-discovery-message-count {
    type yang:zero-based-counter64;
    config false;
    description
        "Number of AMT relay discovery messages sent
        on the interface.";
}
leaf relay-advertisement-message-count {
    type yang:zero-based-counter64;
    config false;
    description
        "Number of AMT relay advertisement messages received

```

Commenté [MB43]: Same as above

```

        on the interface.";
    }
leaf request-message-count {
    type yang:zero-based-counter64;
    config false;
    description
        "Number of AMT membership request messages sent
         on the interface.";
}
leaf membership-query-message-count {
    type yang:zero-based-counter64;
    config false;
    description
        "Number of AMT membership query messages received
         on the interface.";
}
leaf membership-update-message-count {
    type yang:zero-based-counter64;
    config false;
    description
        "Number of AMT membership update messages sent
         on the interface.";
}
}
}
container gateway-message-statistics {
    config false;
    description
        "Message statistics of AMT Gateway.";
    container received {
        description
            "Received message statistics of AMT Gateway.";
        leaf relay-advertisement {
            type yang:zero-based-counter64;
            description
                "Number of AMT relay advertisement messages
                 received.";
        }
        leaf membership-query {
            type yang:zero-based-counter64;
            description
                "Number of AMT membership query messages
                 received.";
        }
    }
    container sent {
        description
            "Sent message statistics of AMT Gateway.";
        leaf relay-discovery {
            type yang:zero-based-counter64;
            description
                "Number of AMT relay discovery messages sent.";
        }
        leaf request {
            type yang:zero-based-counter64;
            description
                "Number of AMT membership request messages sent.";
        }
    }
}

```

```

        leaf membership-update {
            type yang:zero-based-counter64;
            description
                "Number of AMT membership update messages sent.";
        }
        leaf teardown {
            type yang:zero-based-counter64;
            description
                "Number of AMT teardown messages sent.";
        }
    }
}
} // gateway
} // amt
} // augment
}
<CODE ENDS>

```

4. Data Model Example

This section presents a simple and illustrative example of how to configure AMT.

The example is represented in XML [W3C.REC-xml-20081126].

~~The following is Figure X shows~~ – an example configuration for an AMT relay service.

This example configures the protocol address family (IPv4), secret key timeout (120 minutes), and tunnel limit (10) for AMT relay function. In addition, the AMT anycast prefix is set to 10.10.10.10/32 and the AMT local address is configured to 10.255.112.201.

```

<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
        <control-plane-protocols>
            <amt xmlns="urn:ietf:params:xml:ns:yang:ietf-amt">
                <relay>
                    <relay-addresses>
                        <relay-address>
                            <family>ipv4</family>
                            <anycast-prefix>10.10.10.10/32</anycast-prefix>
                            <local-address>10.255.112.201</local-address>
                        </relay-address>
                    </relay-addresses>
                    <tunnel-limit>12</tunnel-limit>
                    <secret-key-timeout>120</secret-key-timeout>
                </relay>
            </amt>
        </control-plane-protocols>
    </routing>
</config>

```

~~Figure X: XXXX~~

5. Security Considerations

Commenté [MB44]: Please move this to an appendix as this is informative.

Commenté [MB45]: Please use documentation addresses in rfc5737

Commenté [MB46]: Please add an example with IPv6
Commenté [MB47]: For readers convenience, I suggest to use JSON?

Commenté [MB48]: Please update to follow the template at <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-28#name-security-considerations-sect>

This section is modeled after the template described in Section 3.7 of [I-D.ietf-netmod].

a mis en forme : Anglais (États-Unis)

The "ietf-amt" YANG module defines a data model that is designed to be accessed via YANG-based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. These protocols have to use a secure transport layer (e.g., SSH [RFC4252], TLS [RFC8446], and QUIC [RFC9000]) and have to use mutual authentication.

The Network Configuration Access Control Model (NACM) [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:

Under /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/:

amt/relay/relay-addresses/relay-address

This subtree specifies the IPv4 or IPv6 address information for an AMT relay. Modifying the configuration may cause the AMT tunnel to be torn down or established.

a mis en forme : Retrait : Gauche : 1,25 cm

Commenté [MB49]: Fix formatting

amt/relay/relay-dns-resource-records/relay-dns-resource-record

This subtree specifies the DNS ~~resource records~~RR configuration used to discover AMT relays. Modifying this configuration may cause the AMT gateway to discover new AMT relay devices, or fail to discover AMT relay devices.

a mis en forme : Retrait : Gauche : 1,25 cm

amt/gateway/pseudo-interfaces/pseudo-interface

This subtree specifies the parameters of AMT pseudo-interface for an AMT gateway. Modifying this configuration may cause the AMT gateway to establish or tear down tunnels with multiple AMT relays.

a mis en forme : Retrait : Gauche : 1,25 cm

Unauthorized access to any data nodes in these subtrees can adversely affect the AMT subsystem of both the local device and the network. This may lead to network malfunctions, delivery of packets to inappropriate destinations, and other problems.

Some of the readable data nodes in this YANG module may be

considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

Under /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/:

amt/relay

amt/gateway

Unauthorized access to any data nodes in these subtrees can disclose operational state information about the AMT relay or AMT gateway on this device.

a mis en forme : Retrait : Gauche : 1,25 cm

6. IANA Considerations

RFC Ed.: ~~In this section, Please~~ replace all occurrences of 'XXXX' with the actual RFC number (and remove this note).

6.1. IETF XML Registry

The IANA is requested to assign the following URI in the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-amt
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

6.2. YANG Module Names Registry

This document request IANA to register the following YANG module in the "YANG Module Names" registry [RFC6020]:

Name: ietf-amt
Maintained by IANA? N
Namespace: urn:ietf:params:xml:ns:yang:ietf-amt
Prefix: amt
Reference: RFC XXXX

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.

- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC7450] Bumgardner, G., "Automatic Multicast Tunneling", RFC 7450, DOI 10.17487/RFC7450, February 2015, <<https://www.rfc-editor.org/info/rfc7450>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <<https://www.rfc-editor.org/info/rfc8294>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- ~~[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.~~
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.
- [RFC8777] Holland, J., "DNS Reverse IP Automatic Multicast Tunneling (AMT) Discovery", RFC 8210, DOI 10.17487/RFC8777, April 2020, <<https://www.rfc-editor.org/info/rfc8777>>.

7.2. Informative References

- [RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <<https://www.rfc-editor.org/info/rfc4252>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.

[editor.org/info/rfc6991>.](https://www.rfc-editor.org/info/rfc6991)

Commenté [MB50]: To be updated to 9911

- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <<https://www.rfc-editor.org/info/rfc9000>>.
- [W3C.REC-xml-20081126] Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <<https://www.w3.org/TR/2008/REC-xml-20081126>>.

Authors' Addresses

Yisong Liu
China Mobile
China
Email: liuyisong@chinamobile.com

Changwang Lin
New H3C Technologies
China
Email: linchangwang.04414@h3c.com

Zheng(Sandy) Zhang
ZTE Corporation
China
Email: zhang.zheng@zte.com.cn

Xuesong Geng
Huawei Technologies
China
Email: gengxuesong@huawei.com

Vinod Kumar Nagaraj
Juniper Networks

Email: vinkumar@juniper.net

