Abstract

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in user packets while the packets traverse a path between two points in the network. This document defines a YANG module for the IOAM function.

Requirements Language

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

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 December 28, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.
In-situ Operations, Administration, and Maintenance (IOAM) [I-D.ietf-ippm-ioam-data] records OAM information within user packets while the packets traverse a network. The data types and data formats for IOAM data records have been defined in [I-D.ietf-ippm-ioam-data]. The IOAM data can be embedded in many protocol encapsulations such as Network Services Header, Segment Routing, and IPv6 [I-D.brockners-inband-oam-transport].

This document defines a data model for IOAM capabilities using the YANG data modeling language [RFC7950]. This YANG model supports all the three categories of IOAM data, which are Tracing Option, Proof of Transit Option, and Edge-to-Edge Option.
1.1. Tree Diagrams

The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Curly braces 
  "{" and "}" contain names of optional features that make the corresponding node conditional.
- Abbreviations before data node names: "rw" means configuration (read-write), "ro" state data (read-only).
- Symbols after data node names: "?" means an optional node, "!" a container with presence, and "*" denotes a "list" or "leaf-list".
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Design of the IOAM YANG Data Model

2.1. Profiles

The IOAM model is organized as list of profiles as shown in the following figure. Each profile associates with one flow and the corresponding IOAM information.

module: ietf-ioam
  +--rw ioam
    +--rw ioam-profiles
    +--rw admin-config
      |  +--rw enabled?   boolean
    +--rw ioam-profile* [profile-name]
      +--rw profile-name                    string
      +--rw filter
        |  +--rw filter-type?   ioam-filter-type
        |  +--rw acl-name?      -> /acl:acls/acl/name
        +--rw protocol-type?   ioam-protocol-type
      +--rw incremental-tracing-profile {incremental-trace}?
      |  ...
      +--rw preallocated-tracing-profile {preallocated-trace}?
      |  ...
      +--rw pot-profile {proof-of-transit}?
      |  ...
      +--rw e2e-profile {edge-to-edge}?
      ...
The "enabled" is an administrative configuration. When it is set to true, IOAM configuration is enabled for the system. Meanwhile, the IOAM data-plane functionality is enabled.

The "filter" is used to identify a flow, where the IOAM profile can apply. There may be multiple filter types. ACL is the default one.

The IOAM data can be encapsulated into multiple protocols, e.g., IPv6 [RFC8200], Geneve [I-D.ietf-nvo3-geneve], VxLAN-GPE [I-D.ietf-nvo3-vxlan-gpe]. The "protocol-type" is used to indicate where the IOAM is applied. For example, if the "protocol-type" is IPv6, the IOAM ingress node will encapsulate the associated flow with the IPv6-IOAM [I-D.brockners-inband-oam-transport] format.

IOAM data includes three usage options with four encapsulation types, i.e., incremental tracing data, preallocated tracing data, prove of transit data and end to end data. In practice, multiple IOAM data types can be encapsulated into the same IOAM header. The "ioam-profile" contains a set of sub-profiles, each of which relates to one encapsulation type. The configured object may not support all the sub-profiles. The supported sub-profiles are indicated by 4 defined features, i.e., "incremental-trace", "preallocated-trace", "proof-of-transit", "edge-to-edge".

2.2. Preallocated Tracing Profile

The IOAM tracing data is expected to be collected at every node that a packet traverses to ensure visibility into the entire path a packet takes within an IOAM domain. The preallocated tracing option will create pre-allocated space for each node to populate its information. The "preallocated-tracing-profile" contains the detailed information for the preallocated tracing data. The information includes:

- enabled: indicates whether the preallocated tracing profile is enabled.

- node-action: indicates the operation (e.g., encapsulate IOAM header, transit the IOAM data, or decapsulate IOAM header) applied to the dedicated flow.

- use-namespace: indicate the namespace used for the trace types.

- trace-type: indicates the per-hop data to be captured by the IOAM enabled nodes and included in the node data list.

- Loopback mode is used to send a copy of a packet back towards the source.
2.3. Incremental Tracing Profile

The incremental tracing option contains a variable node data fields where each node allocates and pushes its node data immediately following the option header. The "incremental-tracing-profile" contains the detailed information for the incremental tracing data. The detailed information is the same as the Preallocated Tracing Profile, but with one more variable, "max-length", which restricts the length of the IOAM header.

```
+-rw incremental-tracing-profile {incremental-trace}?
  +--rw enabled?               boolean
  +--rw node-action?           ioam-node-action
  +--rw trace-types
    |  +--rw use-namespace?   ioam-namespace
    |  +--rw trace-type*      ioam-trace-type
  +--rw enable-loopback-mode?  boolean
  +--rw max-length?            uint32
```

2.4. Proof of Transit Profile

The IOAM Proof of Transit data is to support the path or service function chain verification use cases. The "pot-profile" contains the detailed information for the prove of transit data. The detailed information are described in [I-D.brockners-proof-of-transit].

```
+-rw pot-profile {proof-of-transit}?
  +--rw enabled?               boolean
  +--rw active-profile-index?  pot:profile-index-range
  +--rw pot-profile-list* [pot-profile-index]
    +--rw pot-profile-index  profile-index-range
    +--rw prime-number       uint64
    +--rw secret-share        uint64
    +--rw public-polynomial  uint64
    +--rw lpc                 uint64
    +--rw validator?          boolean
    +--rw validator-key?      uint64
    +--rw bitmask?            uint64
```
2.5. Edge to Edge Profile

The IOAM edge to edge option is to carry data that is added by the IOAM encapsulating node and interpreted by IOAM decapsulating node. The "e2e-profile" contains the detailed information for the edge to edge data. The detailed information includes:

- enabled: indicates whether the edge to edge profile is enabled.
- node-action is the same semantic as in Section 2.2.
- use-namespace: indicate the namespace used for the edge to edge types.
- e2e-type indicates data to be carried from the ingress IOAM node to the egress IOAM node.

```.yang
+--rw e2e-profile {edge-to-edge}?
   +--rw enabled? boolean
   +--rw node-action? ioam-node-action
   +--rw e2e-types
      +--rw use-namespace? ioam-namespace
      +--rw e2e-type* ioam-e2e-type
```

3. IOAM YANG Module

```yang
code begins file "ietf-ioam@2018-06-27.yang"
module ietf-ioam {
   yang-version 1.1;
   prefix "ioam";
   import ietf-pot-profile {
      prefix "pot";
   }

   import ietf-access-control-list {
      prefix "acl";
   }

   organization "IETF IPPM (IP Performance Metrics) Working Group";

   contact
   "WG Web: <http://tools.ietf.org/wg/ippm>
   WG List: <ippm@ietf.org>
   Editor: zhoutianran@huawei.com";

   description
```
"This YANG module specifies a vendor-independent data model for the in Situ OAM (iOAM).";

revision 2019-06-26 {
    description "Initial revision."
    reference "draft-zhou-ippm-ioam-yang"
}

/*************************************************************
* FEATURES
*************************************************************/

feature incremental-trace {
    description "This feature indicated that the incremental tracing mode is supported";
}

feature preallocated-trace {
    description "This feature indicated that the preallocated tracing mode is supported";
}

feature proof-of-transit {
    description "This feature indicated that the proof of transit mode is supported";
}

feature edge-to-edge {
    description "This feature indicated that the edge to edge mode is supported";
}

/*************************************************************
* IDENTITIES
*************************************************************/

identity base-filter {
    description "Base identity to represent a filter. A filter is used to specify the flow to apply the iOAM profile. ";
}
identity acl-filter {
  base base-filter;
  description
    "Apply ACL rule to specify the flow.";
}

identity base-protocol {
  description
    "Base identity to represent the carrier protocol. It’s used to
    indicate what layer and protocol the iOAM data is embedded.";
}

identity ipv6-protocol {
  base base-protocol;
  description
    "The described iOAM data is embedded in ipv6 protocol.";
}

identity base-node-action {
  description
    "Base identity to represent the node actions. It’s used to
    indicate what action the node will take.";
}

identity encapsulate {
  base base-node-action;
  description
    "indicate the node is to encapsulate the iOAM packet";
}

identity transit {
  base base-node-action;
  description
    "indicate the node is to transit the iOAM packet";
}

identity decapsulate {
  base base-node-action;
  description
    "indicate the node is to decapsulate the iOAM packet";
}

identity base-trace-type {
  description
    "Base identity to represent trace types";
}

identity trace-hop-lim-node-id {

base base-trace-type;
description
  "indicates presence of Hop_Lim and node_id in the
  node data.";
}

identity trace-if-id {
  base base-trace-type;
description
  "indicates presence of ingress_if_id and egress_if_id in the
  node data.";
}

identity trace-timestamp-seconds {
  base base-trace-type;
description
  "indicates presence of time stamp seconds in the node data.";
}

identity trace-timestamp-nanoseconds {
  base base-trace-type;
description
  "indicates presence of time stamp nanoseconds in the node data.";
}

identity trace-transit-delay {
  base base-trace-type;
description
  "indicates presence of transit delay in the node data.";
}

identity trace-namespace-data {
  base base-trace-type;
description
  "indicates presence of namespace specific data (short format)
  in the node data.";
}

identity trace-queue-depth {
  base base-trace-type;
description
  "indicates presence of queue depth in the node data.";
}

identity trace-opaque-state-snapshot {
  base base-trace-type;
description
  "indicates presence of variable length Opaque State Snapshot
field."
}

identity trace-hop-lim-node-id-wide {
  base base-trace-type;
  description
    "indicates presence of Hop_Lim and node_id wide in the
    node data.";
}

identity trace-if-id-wide {
  base base-trace-type;
  description
    "indicates presence of ingress_if_id and egress_if_id wide in
    the node data.";
}

identity trace-namespace-data-wide {
  base base-trace-type;
  description
    "indicates presence of namespace specific data in wide format
    in the node data.";
}

identity trace-buffer-occupancy {
  base base-trace-type;
  description
    "indicates presence of buffer occupancy in the node data.";
}

identity trace-checksum-complement {
  base base-trace-type;
  description
    "indicates presence of the Checksum Complement node data.";
}

identity base-pot-type {
  description
    "Base identity to represent pot types";
}

identity pot-bytes-16 {
  base base-pot-type;
  description
    "POT data is a 16 Octet field.";
}

identity base-e2e-type {
typedef ioam-filter-type {
  type identityref {
    base base-filter;
  }
}
typedef ioam-protocol-type {
  type identityref {
    base base-protocol;
  }
  description
    "Specifies a known type of carrier protocol for the iOAM data.";}

typedef ioam-node-action {
  type identityref {
    base base-node-action;
  }
  description
    "Specifies a known type of node action.";}

typedef ioam-trace-type {
  type identityref {
    base base-trace-type;
  }
  description
    "Specifies a known trace type.";}

typedef ioam-pot-type {
  type identityref {
    base base-pot-type;
  }
  description
    "Specifies a known pot type.";}

typedef ioam-e2e-type {
  type identityref {
    base base-e2e-type;
  }
  description
    "Specifies a known e2e type.";}

typedef ioam-namespace {
  type identityref {
    base base-namespace;
grouping ioam-filter {
    description "A grouping for iOAM filter definition";
    leaf filter-type {
        type ioam-filter-type;
        description "filter type";
    }
    leaf acl-name {
        when ".../filter-type = 'acl-filter'";
        type leafref {
            path "/acl:acls/acl:acl/acl:name";
        }
        description "Access Control List name.";
    }
}

grouping ioam-incremental-tracing-profile {
    description "A grouping for incremental tracing profile.";
    leaf node-action {
        type ioam-node-action;
        description "node action";
    }
    container trace-types {
        when ".../node-action = 'encapsulate'";
        description "the list of trace types for encapsulate";
        leaf use-namespace {
            type ioam-namespace;
            description "the namespace used for the encapsulation";
        }
        leaf-list trace-type {
            type ioam-trace-type;
        }
    }
}
description
"The trace type is only defined at the encapsulation node.";
}
}

leaf enable-loopback-mode {
  when "/node-action = 'encapsulate'";
  type boolean;
  default false;
  description
  "Loopback mode is used to send a copy of a packet back towards
  the source. The loopback mode is only defined at the
  encapsulation node."
}

leaf max-length {
  when "/node-action = 'encapsulate'";
  type uint32;
  description
  "This field specifies the maximum length of the node data list
  in octets. The max-length is only defined at the
  encapsulation node. And it’s only used for the incremental
  tracing mode."
}

grouping ioam-preallocated-tracing-profile {
  description
  "A grouping for incremental tracing profile."

  leaf node-action {
    type ioam-node-action;
    description "node action"
  }

  container trace-types {
    when "/node-action = 'encapsulate'";
    description
    "the list of trace types for encapsulate"

    leaf use-namespace {
      type ioam-namespace;
      description
      "the namespace used for the encapsulation"
    }

    leaf-list trace-type {
      type ioam-trace-type;
    }
  }
}
description
 "The trace type is only defined at the encapsulation node.";
 }
}

leaf enable-loopback-mode {
   when "../node-action = 'encapsulate’";
   type boolean;
   default false;
   description
   "Loopback mode is used to send a copy of a packet back towards
   the source. The loopback mode is only defined at the
   encapsulation node.”;
 }
}

grouping ioam-e2e-profile {
   description
   "A grouping for tracing profile.”;

   leaf node-action {
      type ioam-node-action;
      description
      "indicate how the node act for this profile”;
   }

   container e2e-types {
      when "../node-action = ‘encapsulate’";
      description
      "the list of e2e types for encapsulate”;

      leaf use-namespace {
         type ioam-namespace;
         description
         "the namespace used for the encapsulation”;
      }

      leaf-list e2e-type {
         type ioam-e2e-type;
         description
         "The e2e type is only defined at the encapsulation node.”;
      }
   }
}

grouping ioam-admin-config {
   description
   "IOAM top-level administrative configuration.”;

leaf enabled {
  type boolean;
  default false;
  description "When true, IOAM configuration is enabled for the system. Meanwhile, the IOAM data-plane functionality is enabled.";
}

/*
 * DATA NODES
 */

container ioam {
  description "iOAM top level container";
}

container ioam-profiles {
  description "Contains a list of iOAM profiles.";
}

container admin-config {
  description "Contains all the administrative configurations related to the IOAM functionalities and all the IOAM profiles."
  uses ioam-admin-config;
}

list ioam-profile {
  key "profile-name";
  ordered-by user;
  description "A list of iOAM profiles that configured on the node.";
  leaf profile-name {
    type string;
    mandatory true;
    description "Unique identifier for each iOAM profile";
  }
}

container filter {
  uses ioam-filter;
  description "The filter which is used to indicate the flow to apply iOAM.";
}
leaf protocol-type {
    type ioam-protocol-type;
    description "This item is used to indicate the carrier protocol where
the IOAM is applied.";
}

container incremental-tracing-profile {
    if-feature incremental-trace;
    description "describe the profile for incremental tracing option";
    leaf enabled {
        type boolean;
        default false;
        description "When true, apply incremental tracing option to the
specified flow identified by the filter.";
    }
    uses ioam-incremental-tracing-profile;
}

container preallocated-tracing-profile {
    if-feature preallocated-trace;
    description "describe the profile for preallocated tracing option";
    leaf enabled {
        type boolean;
        default false;
        description "When true, apply preallocated tracing option to the
specified flow identified by the following filter.";
    }
    uses ioam-preallocated-tracing-profile;
}

container pot-profile {
    if-feature proof-of-transit;
    description "describe the profile for pot option";
    leaf enabled {
        type boolean;
        default false;
        description
    }
}
"When true, apply Proof of Transit option to the specified flow identified by the following filter."

leaf active-profile-index {
  type pot:profile-index-range;
  description
  "Proof of transit profile index that is currently active. Will be set in the first hop of the path or chain. Other nodes will not use this field."
}

uses pot:pot-profile;

container e2e-profile {
  if-feature edge-to-edge;
  description
  "describe the profile for e2e option"

  leaf enabled {
    type boolean;
    default false;
    description
    "When true, apply End to end option to the specified flow identified by the following filter."
  }

  uses ioam-e2e-profile;
}

<CODE ENDS>

4. 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 [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a
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:

- /ioam/ioam-profiles/admin-config

The items in the container above include the top level administrative configurations related to the IOAM functionalities and all the IOAM profiles. Unexpected changes to these items could lead to the IOAM function disruption and/or misbehavior of all the IOAM profiles.

- /ioam/ioam-profiles/ioam-profile

The entries in the list above include the whole IOAM profile configurations which indirectly create or modify the device configurations. Unexpected changes to these entries could lead to the mistake of the IOAM behavior for the corresponding flows.

5. IANA Considerations

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

IANA is requested to assign a new URI from the IETF XML Registry [RFC3688]. The following URI is suggested:

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

This document also requests a new YANG module name in the YANG Module Names registry [RFC7950] with the following suggestion:

  name: ietf-ioam
  prefix: ioam
  reference: RFC XXXX
6. Acknowledgements

For their valuable comments, discussions, and feedback, we wish to acknowledge Greg Mirsky and Reshad Rahman.

7. References

7.1. Normative References


7.2. Informative References

[I-D.brockners-inband-oam-transport]

[I-D.brockners-proof-of-transit]

[I-D.ietf-ippm-ioam-data]

[I-D.ietf-nvo3-geneve]

[I-D.ietf-nvo3-vxlan-gpe]

Authors’ Addresses

Tianran Zhou
Huawei
156 Beiqing Rd.
Beijing  100095
China

Email: zhoutianran@huawei.com