A YANG Data Model for In-Situ OAM

Abstract

In-situ Operations, Administration, and Maintenance (IOAM) is an example of an on-path hybrid measurement method. IOAM defines a method to produce operational and telemetry information that may be exported using the in-band or out-of-band method. RFC9197 and RFC9326 discuss the data fields and associated data types for IOAM. This document defines a YANG module for the configuration of IOAM 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 2 September 2024.

Copyright Notice

Copyright (c) 2024 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
1. Introduction

In-situ Operations, Administration, and Maintenance (IOAM) is an example of an on-path hybrid measurement method. IOAM defines a method to produce operational and telemetry information that may be exported using the in-band or out-of-band method. The data types and data formats for IOAM data records have been defined in [RFC9197] and [RFC9326]. The IOAM data can be embedded in many protocol encapsulations such as Network Services Header (NSH) and IPv6.

This document defines a data model for the configuration of IOAM capabilities using the YANG data modeling language [RFC7950]. This YANG model supports five IOAM options, which are:

* **Incremental Tracing Option** [RFC9197]

* **Pre-allocated Tracing Option** [RFC9197]

* **Direct Export Option** [RFC9326]

* **Proof of Transit (PoT) Option** [RFC9197]
2. Conventions used in this document

The keywords "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 BCP14, [RFC2119], [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in [RFC7950] and are used in this specification:

* augment
* data model
* data node

The terminology for describing YANG data models is found in [RFC7950].

2.1. Tree Diagrams

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

3. Design of the IOAM YANG Data Model

3.1. Overview

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.
The "info" is a container for all the read-only information that assists monitoring systems in the interpretation of the IOAM data.

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 [RFC8519] is a common way to specify a flow. Each IOAM profile can associate with an ACE(Access Control Entry). IOAM actions MUST be driven by the accepted packets, when the matched ACE "forwarding" action is "accept".

The IOAM data can be encapsulated into multiple protocols, e.g., IPv6 [RFC9486] and NSH [RFC9452]. 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 [RFC9486] format.

In this document, IOAM data includes five encapsulation types, i.e., incremental tracing data, preallocated tracing data, direct export data, proof of transit data and end to end data. In practice,
multiple IOAM data types can be encapsulated into the same IOAM header. The "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 5 defined features, i.e., "incremental-trace", "preallocated-trace", "direct-export", "proof-of-transit" and "edge-to-edge".

This document uses the **Access Control List YANG module** [RFC8519], the **Interfaces YANG module** [RFC8343] and the **LIME Time Types YANG module** [RFC8532].

The YANG data model in this document conform to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

### 3.2. 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:

* 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: indicates 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.

* max-length: specifies the maximum length of the node data list in octets. The max-length is only defined at the encapsulation node.

```
+--rw preallocated-tracing-profile {preallocated-trace}?
    +--rw node-action?             ioam-node-action
    +--rw trace-types
        |   +--rw use-namespace?   ioam-namespace
        |   +--rw trace-type*   ioam-trace-type
        +--rw max-length?             uint32
```

### 3.3. 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.

3.4. Direct Export Profile

The direct export option is used as a trigger for IOAM data to be directly exported or locally aggregated without being pushed into in-flight data packets. The "direct-export-profile" contains the detailed information for the direct export data. The detailed information is the same as the Preallocated Tracing Profile, but with two more optional variables:

*flow-id: is used to correlate the exported data of the same flow from multiple nodes and from multiple packets.

*enable-sequence-number: indicates whether the sequence number is used in the direct export option.

3.5. Proof of Transit Profile

The IOAM Proof of Transit data is intended to support the path or service function chain verification use cases. The "pot-profile" is intended to contain the detailed information for the proof of transit data. "use-namespace" indicates the namespace used for the POT types. "pot-type" indicates a particular POT variant that specifies the POT data that is included. There may be several POT types, which have different configuration data. To align with [RFC9197], this document only defines IOAM POT type 0. User need to augment this module for the configuration of a specific POT type.
3.6. 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:

*node-action is the same semantic as in Section 3.2.  
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.

```plaintext
+-rw e2e-profile {edge-to-edge}?  
  +-rw node-action?   ioam-node-action  
  +-rw e2e-types  
    +-rw use-namespace?   ioam-namespace  
    +-rw e2e-type*       ioam-e2e-type
```

4. IOAM YANG Module
<CODE BEGINS> file "ietf-ioam@2024-03-01.yang"

module ietf-ioam {
    yang-version 1.1;
    prefix "ioam";

    import ietf-access-control-list {
        prefix "acl";
        reference
            "RFC 8519: YANG Data Model for Network Access Control Lists (ACLs)";
    }

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

    import ietf-lime-time-types {
        prefix "lime";
        reference
            "RFC 8532: Generic YANG Data Model for the Management of Operations, Administration, and Maintenance (OAM) Protocols That Use Connectionless Communications";
    }

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

    contact
        "WG Web: <https://datatracker.ietf.org/wg/ippm>
            WG List: <ippm@ietf.org>
            Editor: zhoutianran@huawei.com
            Editor: james.n.guichard@futurewei.com
            Editor: fbrockne@cisco.com
            Editor: srihari@cisco.com"

    description
        "This YANG module specifies a vendor-independent data model for the In Situ OAM (IOAM).

        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.

        Copyright (c) 2024 IETF Trust and the persons identified as
/* FEATURES */

feature incremental-trace
{
    description "This feature indicated that the incremental tracing option is supported.";
    reference "RFC 9197: Data Fields for In-situ OAM";
}

feature preallocated-trace
{
    description "This feature indicated that the preallocated tracing option is supported.";
    reference "RFC 9197: Data Fields for In-situ OAM";
}

feature direct-export
{
    description "This feature indicated that the direct export option is supported.";
    reference "RFC 9326: In-situ OAM Direct Exporting";
}

feature proof-of-transit
{
    description "This feature indicated that the proof of transit option is
supported);
    reference "RFC 9197: Data Fields for In-situ OAM";
}

feature edge-to-edge
{
    description
        "This feature indicated that the edge-to-edge option is
        supported.";
    reference "RFC 9197: Data Fields for In-situ OAM";
}

/*
* IDENTITIES
*/
identity 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 filter;
    description
        "Apply ACL rules to specify the flow.";
}

identity 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 {
    base protocol;
    description
        "The described IOAM data is embedded in IPv6 protocol.";
    reference
        "RFC 9486: In-situ OAM IPv6 Options";
}

identity nsh {
    base protocol;
    description
        "The described IOAM data is embedded in NSH.";
    reference
        "RFC 9452: Network Service Header (NSH)
         Encapsulation for In-situ OAM (IOAM) Data";
identity node-action {
    description
        "Base identity to represent the node actions. It's used to indicate what action the node will take."
}

identity action-encapsulate {
    base node-action;
    description
        "It indicates the node is to encapsulate the IOAM packet"
}

identity action-decapsulate {
    base node-action;
    description
        "It indicates the node is to decapsulate the IOAM packet"
}

identity action-transit {
    base node-action;
    description
        "It indicates the node is to transit the IOAM packet"
}

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

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

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

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

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

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

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

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

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

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

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

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

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

identity pot-type {
  description
    "Base identity to represent Proof of Transit (PoT) types.";
}

identity pot-type-0 {
  base pot-type;
  description
    "The IOAM POT Type field value is 0, and POT data is a 16 Octet field to carry data associated to POT procedures.";
}

identity e2e-type {
  description
    "Base identity to represent edge-to-edge types.";
}

identity e2e-seq-num-64 {
  base e2e-type;
  description
    "It indicates presence of a 64-bit sequence number.";
}

identity e2e-seq-num-32 {
  base e2e-type;
  description
    "It indicates the presence of a 32-bit sequence number.";
}

identity e2e-timestamp-seconds {
  base e2e-type;
  description
    "It indicates the presence of timestamp seconds representing
the time at which the packet entered the IOAM-domain.

identity e2e-timestamp-fraction {
    base e2e-type;
    description
        "It indicates the presence of timestamp fraction representing
        the time at which the packet entered the IOAM-domain."
}

identity namespace {
    description
        "Base identity to represent the Namespace-ID."
}

identity default-namespace {
    base namespace;
    description
        "The Namespace-ID value of 0x0000 is defined as the
         Default-Namespace-ID and MUST be known to all the nodes
         implementing IOAM."
}

/*
* TYPE DEFINITIONS
*/
typedef ioam-filter-type {
    type identityref {
        base filter;
    }
    description
        "It specifies a known type of filter."
}

typedef ioam-protocol-type {
    type identityref {
        base protocol;
    }
    description
        "It specifies a known type of carrier protocol for the IOAM
        data."
}

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

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

typedef ioam-e2e-type {
    type identityref {
        base e2e-type;
    }
    description
        "It specifies a known edge-to-edge type."
}

typedef ioam-namespace {
    type identityref {
        base namespace;
    }
    description
        "It specifies the supported namespace."
}

/*
 * GROUP DEFINITIONS
 */
grouping ioam-filter {
    description "A grouping for IOAM filter definition";

    leaf filter-type {
        type ioam-filter-type;
        description "filter type";
    }

    leaf ace-name {
        when "derived-from-or-self(../filter-type, 'ioam:acl-filter')";
        type leafref {
            path "'/acl:acls/acl:acl/acl:aces/acl:ace/acl:name'";
        }
    }
}
description "The Access Control Entry name is used to refer to an ACL specification.";
}
}
grouping encap-tracing {

description "A grouping for the generic configuration for tracing profile.";

container trace-types {

description "It indicates the list of trace types for encapsulation.";

leaf use-namespace {

type ioam-namespace;

default default-namespace;

description "It indicates the name space used for encapsulation.";
}

leaf-list trace-type {

type ioam-trace-type;

description "The trace type is only defined at the encapsulation node.";
}
}

leaf max-length {

description "This field specifies the maximum length of the node data list in octets. The max-length is only defined at the encapsulation node.";
}
}

grouping ioam-incremental-tracing-profile {

description "A grouping for incremental tracing profile.";

leaf node-action {

type ioam-node-action;

default action-transit;
description
  "This object indicates the action the node need to take, e.g. encapsulation."
}

uses encap-tracing {
  when "derived-from-or-self(node-action, 'ioam:action-encapsulate')"
}
}


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

  leaf node-action {
    type ioam-node-action;
    default action-transit;
    description
      "This object indicates the action the node need to take, e.g. encapsulation."
  }

  uses encap-tracing {
    when "derived-from-or-self(node-action, 'ioam:action-encapsulate')"
  }
}


grouping ioam-direct-export-profile {
  description
  "A grouping for direct export profile."

  leaf node-action {
    type ioam-node-action;
    default action-transit;
    description
      "This object indicates the action the node need to take, e.g. encapsulation."
  }

  uses encap-tracing {
    when "derived-from-or-self(node-action, 'ioam:action-encapsulate')"
  }
}

leaf flow-id {
  when "derived-from-or-self(../node-action, 'ioam:action-encapsulate')"
  type uint32;
}
description
"A 32-bit flow identifier. The field is set at the
encapsulating node. The Flow ID can be uniformly assigned
by a central controller or algorithmically generated by the
encapsulating node. The latter approach cannot guarantee
the uniqueness of Flow ID, yet the conflict probability is
small due to the large Flow ID space. flow-id is used to
correlate the exported data of the same flow from multiple
nodes and from multiple packets."

leaf enable-sequence-number {
   when "derived-from-or-self(../node-action,
       'ioam:action-encapsulate')"
   type boolean;
   default false;
   description
   "This boolean value indicates whether the sequence number is
used in the direct export option 32-bit flow identifier. If
this value is true, the sequence number is used. By default,
it's turned off."
}

grouping ioam-e2e-profile {
   description
   "A grouping for edge-to-edge profile."

   leaf node-action {
      type ioam-node-action;
      default action-transit;
      description
      "This object indicates the action the node need to
take, e.g. encapsulation."
   }

   container e2e-types {
      when "derived-from-or-self(../node-action,
          'ioam:action-encapsulate')"
      description
      "It indicates the list of edge-to-edge types for
capsulation."

      leaf use-namespace {
         type ioam-namespace;
         default default-namespace;
         description
         "It indicates the name space used for encapsulation."
      }
   }
}
leaf-list e2e-type {
  type ioam-e2e-type;
  description
    "The edge-to-edge 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
      "This object is to control the availability of configuration.
      It MUST be true before anything in the
      /ioam/profiles/profile subtree can be edited.
      If false, any configuration in place is not used.";
  }
}

/*
 * DATA NODES
 */

container ioam {
  description "IOAM top level container";

  container info {
    config false;
    description
      "Describes information such as units or timestamp format that
      assists monitoring systems in the interpretation of the IOAM
      data.";

    leaf timestamp-type {
      type identityref {
        base lime:timestamp-type;
      } 
      description
        "Type of timestamp, such as Truncated PTP or NTP.";
    }

    list available-interface {
      key "if-name";
    }
  }
}
description
  "A list of available interfaces that support IOAM.";
leaf if-name {
  type if:interface-ref;
  description "This is a reference to the Interface name.";
}
}

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

  uses ioam-admin-config;
}

container profiles {
  description
  "Contains a list of IOAM profiles.";

  list profile {
    key "profile-name";
    description
      "A list of IOAM profiles that configured on the node.
      There is no mandatory type of profile (e.g.,
      incremental-trace, preallocated-trace.) in the list.
      But at least one profile should be added.";

    leaf profile-name {
      type string{
        length "1..300";
      }
      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;
    presence "Enables incremental tracing option.";
    description
        "It describes the profile for incremental tracing option.";

    uses ioam-incremental-tracing-profile;
}

container preallocated-tracing-profile {
    if-feature preallocated-trace;
    presence "Enables preallocated tracing option.";
    description
        "It describes the profile for preallocated tracing option.";

    uses ioam-preallocated-tracing-profile;
}

container direct-export-profile {
    if-feature direct-export;
    presence "Enables direct-export option.";
    description
        "It describes the profile for direct-export option";

    uses ioam-direct-export-profile;
}

container pot-profile {
    if-feature proof-of-transit;
    presence "Enables Proof of Transit option.";
    description
        "It describes the profile for PoT option.";

    leaf use-namespace {
        type ioam-namesapce;
        default default-namespace;
        description
            "It indicates the namespace used for the POT types.";
    }

    leaf pot-type {
        type ioam-pot-type;
        description
            "The type of a particular POT variant that specifies the POT data that is included.";
    }
}
container e2e-profile {
  if-feature edge-to-edge;
  presence "Enables edge-to-edge option.";
  description
    "It describes the profile for edge-to-edge option.";

  uses ioam-e2e-profile;
}
}
5. 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 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:

*/ioam/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/profiles/profile: The entries in the list above include the whole IOAM profile configurations. Unexpected changes to these entries could lead to the mistake of the IOAM behavior for the corresponding flows. Consequently, it will impact the performance monitoring, data analytics, and the associated reaction to network services.

Some readable data nodes in these YANG modules 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:

*/ioam/profiles/profile: The information contained in this subtree might give information about the services deployed for the customers. For instance, a customer might be given access to monitor their services status. In that example, the customer access should be restricted to nodes representing their services so as not to divulge information about the underlying network structure or services.
6. 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
```

7. Acknowledgements

For their valuable comments, discussions, and feedback, we wish to acknowledge Greg Mirsky, Reshad Rahman, Tom Petch, Mickey Spiegel, Thomas Graf, Alex Huang Feng and Justin Iurman.

8. Normative References


[ RFC9326] Song, H., Gafni, B., Brockners, F., Bhandari, S., and T. Mizrahi, "In Situ Operations, Administration, and
Appendix A. An Example of Incremental Tracing Profile

An example of incremental tracing profile is depicted in the following figure. This configuration is received by an IOAM ingress node. This node encapsulates the IOAM data in IPv6 Hop-by-Hop option header. The trace type indicates that each on path node need to capture the transit delay, and add to the IOAM node data list. The incremental tracing data space is variable, however, the node data list must not exceed 512 bytes.
Appendix B. An Example of Pre-allocated Tracing Profile

An example of pre-allocated tracing profile is depicted in the following figure. This configuration is received by an IOAM ingress node. This node firstly identifies the target flow by using ACL "test-acl", and then encapsulates the IOAM data in the NSH header. The trace type indicates that each on path node need to capture the name space specific data in the short format, and add to the IOAM node data list. This node preallocates the node data list in the packect with 512 bytes.
Appendix C. An Example of Direct Export Profile

An example of direct export profile is depicted in the following figure. This configuration is received by an IOAM egress node. This node detects the IOAM direct export option in the IPv6 extension header, and removes the option to clean all the IOAM data.
Appendix D. An Example of Proof of Transit Profile

The following figure is a simple example of POT option. This configuration indicates the node to apply POT type 0 with IPv6 encapsulation.

```xml
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <ioam xmlns="urn:ietf:params:xml:ns:yang:ietf-ioam">
        <admin-config>
          <enabled>true</enabled>
        </admin-config>
        <profiles>
          <profile>
            <profile-name>ietf-test-profile</profile-name>
            <protocol-type>ipv6</protocol-type>
            <direct-export-profile>
              <node-action>action-decapsulate</node-action>
            </direct-export-profile>
          </profile>
        </profiles>
      </ioam>
    </config>
  </edit-config>
</rpc>
```
Appendix E. An Example of Edge-to-Edge Profile

The following figure shows an example of edge-to-edge option. This configuration is received by an IOAM egress node. This node detects the IOAM edge-to-edge option in the IPv6 extension header, and removes the option to clean all the IOAM data. As the IOAM egress node, it may collect the edge-to-edge data and deliver to the data exporting process.

```xml
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <ioam xmlns="urn:ietf:params:xml:ns:yang:ietf-ioam">
        <admin-config>
          <enabled>true</enabled>
        </admin-config>
        <profiles>
          <profile>
            <profile-name>ietf-test-profile</profile-name>
            <protocol-type>ipv6</protocol-type>
            <pot-profile>
              <pot-type>pot-type-0</pot-type>
            </pot-profile>
          </profile>
        </profiles>
      </ioam>
    </config>
  </edit-config>
</rpc>
```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <ioam xmlns="urn:ietf:params:xml:ns:yang:ietf-ioam">
        <admin-config>
          <enabled>true</enabled>
        </admin-config>
        <profiles>
          <profile>
            <profile-name>ietf-test-profile</profile-name>
            <protocol-type>ipv6</protocol-type>
            <e2e-profile>
              <node-action>action-decapsulate</node-action>
            </e2e-profile>
          </profile>
        </profiles>
      </ioam>
    </config>
  </edit-config>
</rpc>

Authors' Addresses

Tianran Zhou
Huawei
156 Beiqing Rd.
Beijing
100095
China

Email: zhoutianran@huawei.com

Jim Guichard
Futurewei
United States of America

Email: james.n.guichard@futurewei.com

Frank Brockners
Cisco Systems
Hansaallee 249, 3rd Floor
40549 Duesseldorf
Germany

Email: fbrockne@cisco.com

Srihari Raghavan
Cisco Systems
Tril Infopark Sez, Ramanujan IT City
Neville Block, 2nd floor, Old Mahabalipuram Road
Chennai 600113
Tamil Nadu
India

Email: srihari@cisco.com