Taps
Internet-Draft
Updates: I-D.ietf-taps-interface (if approved)
Intended status: Standards Track
Expires: September 5, 2019

J. Holland
Akamai Technologies, Inc.

March 04, 2019

A YANG Data Model for a Transport Services API at Endpoints
draft-jholland-taps-api-yang-00

Abstract

This document defines a YANG data model that provides a data structure that can be used to configure an implementation of the Transport Services Interface to establish connections suitable for sending and receiving data over the internet or local networks. This document is intended to supplement or merge with draft-ietf-taps-interface.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 5, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
1.1. A Note On The Use Of YANG 3
2. Examples 3
2.1. Basic Client Connection 3
2.2. Customized Connections 3
2.3. Send and Receive Multicast 4
3. YANG Module 6
4. Security Considerations 11
5. IANA Considerations 11
6. Normative References 11
Appendix A. Future Work 12
Author's Address 13

1. Introduction

This document is an attempt to concretize the properties and objects of the TAPS interface described in [I-D.ietf-taps-interface], under the architecture described in [I-D.ietf-taps-arch].

A TAPS-compliant implementation SHOULD provide a language-appropriate way to configure a PreConnection using YANG instance data for this model, and SHOULD provide an API that outputs the YANG instance data for an established Connection.

An implementation MAY also provide appropriate APIs for directly editing the objects without using YANG. It's RECOMMENDED where possible to use names that mechanically translate to the names in the YANG data model, using capitalization and punctuation conventions as expected for the language of the implementation.

Non-TAPS extensions to API objects that directly edit TAPS properties are RECOMMENDED to include "ext", "EXT", or "Ext" as a prefix to any extension properties or methods, to avoid colliding with future TAPS extensions.

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.

1.1. A Note On The Use Of YANG

Although YANG was originally designed to model data for NETCONF, YANG can be used in other ways as well, as described by Section 4.1 of [RFC7950].

This usage is not primarily targeted at NETCONF, but rather at Application Programming Interfaces for libraries that set up connections for sending and receiving data over the internet. However, use of YANG in this context provides a semantically clear, well defined, extensible, cross-platform method for configuring connection objects suitable for replacing BSD sockets in a wide variety of applications.

2. Examples

2.1. Basic Client Connection

The API is designed to allow defaults to fill out almost everything. This example shows the minimal preconnection configuration input data to open a reliable transfer to example.com, via any supported reliable transport protocol on the default port or ports.

<CODE BEGINS>
```
{
  "ietf-taps-api:preconnection":{
    "remote-endpoint":{
      "hosts":[
        {
          "host":"example.com"
        }
      ]
    }
  }
}
```
<CODE ENDS>

2.2. Customized Connections

In some cases, applications may have explicit preferences, either dynamically inferred from past statistics or configured via system or app preferences of some kind.

This example demonstrates adding constraints on the endpoints when opening a connection. For example, an app that needs to avoid using a local proxy for a specific set of connections might configure those connections to prohibit connecting to local IP addresses:

```
<CODE BEGINS>
{
  "ietf-taps-api:preconnection":{
    "local-endpoint":{
      "addresses":[
        {
          "address":"::1",
          "preference":"prohibit"
        },
        {
          "address":"127.0.0.1",
          "preference":"prohibit"
        }
      ]
    },
    "remote-endpoint":{
      "hosts":[
        {
          "host":"example.com"
        }
      ],
      "ports":[
        {
          "port":"443", "preference":"prefer"
        }
      ]
    }
  }
}
<CODE ENDS>
```

Figure 1: Customized endpoint selection

2.3. Send and Receive Multicast

Sending to a multicast group is the same as any non-reliable, non-ordered connection:

<CODE BEGINS>

```
{
  "ietf-taps-api:preconnection":{
    "remote-endpoint":{
      "hosts":[
        {
          "host":"232.252.0.2",
          "preference":"require"
        }
      ],
      "ports":[
        {
          "port":"30000",
          "preference":"require"
        }
      ]
    },
    "transport-properties":{
      "properties":[
        {
          "property":"reliable",
          "preference":"avoid"
        },
        {
          "property":"preserve-order",
          "preference":"avoid"
        }
      ]
    }
  }
}
```

<CODE ENDS>

Figure 2: PreConnection for Sending Multicast

Receiving multicast is similar. It may use remote-endpoint to specify a source-specific multicast subscription, or exclude it to specify any-source multicast.

```
<CODE BEGINS>
{
  "ietf-taps-api:preconnection":{
    "remote-endpoint":{
      "hosts":[
        {
          "host":"192.0.2.15"
        }
      ]
    },
    "local-endpoint":{
      "addresses":[
        {
          "address":"232.252.0.2",
          "preference":"require"
        }
      ],
      "ports":[
        {
          "port":"30000",
          "preference":"require"
        }
      ]
    },
    "transport-properties":{
      "properties":[
        {
          "property":"reliable",
          "preference":"avoid"
        },
        {
          "property":"preserve-order",
          "preference":"avoid"
        }
      ]
    }
  }
}
<CODE ENDS>
```

Figure 3: PreConnection for Source-specific Multicast Receive

3. YANG Module

```
<CODE BEGINS> file "ietf-taps-api@2019-03-04.yang"
module ietf-taps-api {
  yang-version 1.1;
```

```
  namespace "urn:ietf:params:xml:ns:yang:ietf-taps-api";
  prefix "taps";

  import ietf-yang-types {
      prefix "yang";
      reference "[RFC6991] Section 3";
  }

  import ietf-inet-types {
      prefix "inet";
      reference "[RFC6991] Section 4";
  }

  import ietf-interfaces {
      prefix "if";
      reference "[RFC8343] Section 5";
  }

  organization "IETF";

  contact
      "Author:   Jake Holland
                 <mailto:jholland@akamai.com>
      ";

  description
      "This module contains the definition for the TAPS
       interface.

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

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

       This version of this YANG module is part of
       draft-jholland-taps-api-yang,
       see the internet draft itself for full legal notices.";

  revision 2019-02-26 {
      description "Initial revision.";
      reference
        "draft-jholland-taps-api-yang";
  }
```

```
  typedef preference-level {
    type enumeration {
      enum require {
        description
          "select only options providing this property, fail otherwise";
      }
      enum prefer {
        description
          "prefer options providing this property, proceed otherwise";
      }
      enum ignore {
        description
          "cancel any system default preference for this property";
      }
      enum avoid {
        description
          "prefer options not providing the property,
           proceed otherwise";
      }
      enum prohibit {
        description
          "select only options not proiding ths property, fail
           otherwise";
      }
    }
    description
      "This value represents the preference level of a property.";
  }

  container preconnection {
    description "preconnection config for a taps connection";
    container local-endpoint {
      description "local-endpoint config for a preconnection";
      list addresses {
        key "address";

        leaf address {
          type inet:ip-address;
          mandatory true;
          description "address option for the local endpoint";
        }
        leaf preference {
          type preference-level;
          default require;
          description "preference level for the address";
        }
        description
          "list of address constrants for the local endpoint";
```

```
      }

      list ports {
        key "port";

        leaf port {
          type inet:port-number;
          mandatory true;
          description "port option for the local endpoint";
        }
        leaf preference {
          type preference-level;
          default require;
          description "preference level for the port";
        }
        description
          "list of port constrants for the local endpoint";
      }

      list interfaces {
        key "interface";

        leaf interface {
          type if:interface-ref;
          mandatory true;
          description "interface option for the local endpoint";
        }
        leaf preference {
          type preference-level;
          default require;
          description "preference level for the interface";
        }
        description
          "list of interface constrants for the local endpoint";
      }
    }

    container remote-endpoint {
      description "remote-endpoint config for a preconnection";
      list hosts {
        key "host";

        leaf host {
          type inet:host;
          mandatory true;
          description "host property for remote endpoint";
        }
        leaf preference {
```

```
          type preference-level;
          default require;
          description "preference level for the host";
        }
        description "list of host constraints for the remote endpoint";
      }

      list ports {
        key "port";

        leaf port {
          type inet:port-number;
          mandatory true;
          description "port number property for remote endpoint";
        }
        leaf preference {
          type preference-level;
          default require;
          description "preference level for the port";
        }
        description "list of port constraints for the remote endpoint";
      }
    }

    container transport-properties {
      list properties {
        key "property";

        leaf property {
          type enumeration {
            enum reliable {
              description "reliable transport";
            }
            enum per-message-reliable {
              description "per message reliability";
            }
            enum preserve-order {
              description "preserve data ordering";
            }
          }
          mandatory true;
          description
            "transport property for a preconnection";
        }
        leaf preference {
          type preference-level;
          default require;
          description "preference level for the transport property";
```

```
        }
        description "list of transport property constraints.";
      }
      description "Transport properties for the preconnection";
    }
  }
}
<CODE ENDS>
```

Figure 4: TAPS Interface YANG model

4. Security Considerations

This document describes a configuration system for an API that may replace sockets. All security considerations applicable to socket programming should be carefully considered by implementors.

(TBD: surely there is a sane reference, but also fill this out with something less laughable. In particular, enumerate which options should be privileged operations or not to preserve the security of BSD sockets, such as it is. And maybe another layer of restriction recommendations for sandboxed or browser systems.)

5. IANA Considerations

IANA is requested to add the YANG model in Section 3 to the yang-parameters registry.

Field	Value
Name	ietf-taps-api
Namespace	urn:ietf:params:xml:ns:yang:ietf-taps-api
Prefix	taps
Reference	[TBD: this document]

6. Normative References

[I-D.ietf-taps-arch]
Pauly, T., Trammell, B., Brunstrom, A., Fairhurst, G., Perkins, C., Tiesel, P., and C. Wood, "An Architecture for Transport Services", draft-ietf-taps-arch-02 (work in progress), October 2018.

[I-D.ietf-taps-interface]
Trammell, B., Welzl, M., Enghardt, T., Fairhurst, G., Kuehlewind, M., Perkins, C., Tiesel, P., and C. Wood, "An Abstract Application Layer Interface to Transport Services", draft-ietf-taps-interface-02 (work in progress), October 2018.

[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>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[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>.

[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>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

Appendix A. Future Work

TBD if adopted:

- Review [RFC8407] guidelines for YANG authors and reviewers, make sure they are followed.

- Start a reference implementation. No doubt it will highlight many model problems.

- add the rest of the properties in draft-ietf-taps-interface

- many more config examples with use cases

- o Look into providing explicit layers that support some kind of pass-thru, instead of having all properties in big groups of properties.

Author's Address

Jake Holland
Akamai Technologies, Inc.
150 Broadway
Cambridge, MA 02144
United States of America

Email: jakeholland.net@gmail.com