A YANG Data Model for a Transport Services API at Endpoints
draft-jholland-taps-api-yang-03

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 January 8, 2020.

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
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. Tree Diagram

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

module: ietf-taps-api
+-rw preconnection
  +-rw local-endpoints* [id]
    |  +-rw id string
    |  +-rw local-address? inet:ip-address
    |  +-rw local-port? inet:port-number
    |  +-rw stun-info
    |     |  +-rw host? inet:host
    |     |  +-rw port? inet:port-number
    |     |  +-rw identity? string
    |     |  +-rw trust-ca? string
    |     |  +-rw algorithm? identityref
    |     |  +-rw pre-shared-key? string
    |     |  +-rw private-key? string
    |     |  +-rw private-key-callback-handle? string
    |     |  +-rw public-key? string
  +-rw remote-endpoints* [id]
    |  +-rw id string
    |  +-rw remote-host? inet:host
    |  +-rw remote-port? inet:port-number
  +-rw transport-properties
    |  +-rw reliability? preference-level
    |  +-rw preserve-msg-boundaries? preference-level
    |  +-rw per-message-reliability? preference-level
    |  +-rw preserve-order? preference-level
    |  +-rw zero-rtt-msg? preference-level
    |  +-rw multistreaming? preference-level
    |  +-rw per-msg-checksum-len-send? preference-level
    |  +-rw per-msg-checksum-len-recv? preference-level
Tree Diagram

3. Examples

3.1. 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.

```json
{
    "ietf-taps-api:preconnection":{
        "remote-endpoints":{
            "id":"option1",
            "remote-host":"example.com"
        }
    }
}
```

Basic Client Connection
Due to the defaults recommended in (TBD: fix reference) Section 5 of draft-ietf-taps-interface-02, implementations SHOULD treat this basic example equivalently to the same example with the defaults explicitly provided:

```json
{
    "ietf-taps-api:preconnection":{
        "remote-endpoints":{
            "id":"option1",
            "remote-host":"example.com"
        },
        "transport-properties":{
            "reliability":"require",
            "preserve-order":"require",
            "congestion-control":"require"
        }
    }
}
```

Basic Client Connection Explicitly Declaring Defaults

### 3.2. 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.

These examples demonstrate adding constraints on the endpoints when opening a connection.

#### 3.2.1. Require Wi-Fi

This example demonstrates an app that requires the use of a wireless interface:
{  
    "ietf-taps-api:preconnection":{
        "remote-endpoints":{
            "id":"option1",
            "remote-host":"example.com"
        },
        "transport-properties":{
            "interface":{
                "preference":"avoid",
                "value":"iana-if-type:capwapDot11Profile"
            }
        }
    }
}

Figure 1: Customized to require wireless.

3.3. Send and Receive Multicast

Sending to a multicast group is the same as any non-reliable, non-ordered connection:
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.
Figure 3: PreConnection for Source-specific Multicast Receive

3.4. Connecting Through a Stun Server

STUN server connections are a local-endpoint property, and can be configured the same way.


```yaml
{
  "ietf-taps-api:preconnection":{
    "remote-endpoints":{
      "id":"option1",
      "remote-host":"example.com"
    }
  },
  "local-endpoints":{
    "id":"option1",
    "stun-info":{
      "host":"203.0.113.4",
      "port":"10000",
      "identity":"user@mail.example.com",
      "pre-shared-key":"<pre-shared-key>"
    }
  }
}
```

Figure 4: Connect through a STUN server

4. YANG Module

<CODE BEGINS> file ietf-taps-api@2019-07-07.yang
module ietf-taps-api {
  yang-version 1.1;

  prefix "taps";

  import ietf-inet-types {
    prefix "inet";
    reference "RFC 6991 Section 4";
  }

  import ietf-interfaces {
    prefix "if";
    reference "RFC 8343 Section 5";
  }

  import iana-if-type {
    prefix "ianaift";
    reference "RFC 7224 Section 2";
  }
</CODE BEGINS>
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 providing this property, fail otherwise";
}

description
"This value represents the preference level of a property."
}

identity transport-property-type {
    description "Base identity for transport properties";
}

identity provisioning-domain {
    base transport-property-type;
    description "Base for provisioning domain. TBD: add relevant provisioning domain types";
    reference "RFC 7556: Multiple Provisioning Domain Architecture";
}

identity transport-protocol {
    base transport-property-type;
    description "Identity for a transport selection. TBD: finish the rest of the protocols in
    https://tools.ietf.org/html/rfc8095#section-3.1
    maybe add quic, maybe use as external augment demo.
    note: this isn't in taps-interface, but is available in
e.g. NEAT-project/neat/examples/client.c";
    reference "Section 3 of RFC 8095: Existing Transport Protocols";
}

identity tcp {
    base transport-protocol;
    description "Identity for TCP";
    See also RFC 7414: A Roadmap for TCP Specification Documents.";
}

identity mptcp {
    base transport-protocol;
    description "Identity for MPTCP";
    reference
    "RFC 6824: TCP Extensions for Multipath Operation";
}

identity udp {
    base transport-protocol;
description "Identity for UDP";
  reference "TBD";
}

identity udp-lite {
  base transport-protocol;
  description "Identity for UDP-Lite";
  reference "TBD";
}

identity sctp {
  base transport-protocol;
  description "Identity for SCTP";
  reference "TBD";
}

identity dccp {
  base transport-protocol;
  description "Identity for DCCP";
  reference "TBD";
}

identity tls {
  base transport-protocol;
  description "Identity for TLS";
  reference "TBD";
}

identity rtp {
  base transport-protocol;
  description "Identity for RTP";
  reference "TBD";
}

identity http {
  base transport-protocol;
  description "Identity for HTTP";
  reference "TBD";
}

identity flute {
  base transport-protocol;
  description "Identity for FLUTE";
  reference "TBD";
}

identity norm {
  base transport-protocol;
  description "Identity for NORM";
  reference "TBD";
}

identity icmp {
  base transport-protocol;
  description "Identity for ICMP";
  reference "TBD";
}
identity security-algorithm {
    description "Base identity for security algorithms.";
}

identity cipher-suite {
    base security-algorithm;
    description "Base identity for security cipher suites.";
}

identity signature-algorithm {
    base security-algorithm;
    description "Base identity for security signature algorithms.";
}

identity ed25519 {
    base signature-algorithm;
    description "Identity for ED25519";
}

identity TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 {
    base cipher-suite;
    description "Identity for ECDHE ECDSA with ChaCha20, Poly1305,
        and SHA256";
}

grouping security-credentials {
    description "security credentials";

    leaf identity {
        type string;
        description "identity for security credentials";
    }

    leaf trust-ca {
        type string;
        description "trust-ca for security credentials";
    }

    leaf algorithm {
        type identityref {
            base security-algorithm;
        }
        description "security algorithm for credentials";
    }

    leaf pre-shared-key {
        type string;
        description "pre-shared key for security credentials";
    }

    }
leaf private-key {
  type string;
  description "private key for security credentials";
}

leaf private-key-callback-handle {
  type string;
  description "private key callback handle for externally managed security credentials";
}

leaf public-key {
  type string;
  description "public key for security credentials";
}

container preconnection {
  description "preconnection config for a taps connection";

  list local-endpoints {
    key "id";
    description "list of local endpoints";

    leaf id {
      type string;
      description "id of the local endpoint";
    }

    leaf local-address {
      type inet:ip-address;
      description "ip address of local endpoint";
    }

    leaf local-port {
      type inet:port-number;
      description "port value of an endpoint port";
    }

  }

  container stun-info {
    description "config for the stun server";

    leaf host {
      type inet:host;
      description "stun server host";
    }

    leaf port {

type inet:port-number;
    description "port number for the stun server";
} uses security-credentials;

 Việc TĐ: multicast-subscription: source-specific */

list remote-endpoints {
    key "id";
    description "list of remote endpoints";

    leaf id {
        type string;
        description "id of the remote endpoint";
    }

    leaf remote-host {
        type inet:host;
        description "host value of a remote endpoint";
    }

    leaf remote-port {
        type inet:port-number;
        description "port value of an endpoint port";
    }
}

container transport-properties {
    description "transport property constraints";

    leaf reliability {
        type preference-level;
        default "require";
        description "Section 5.2.1 of draft-ietf-taps-interface-03";
    }

    leaf preserve-msg-boundaries {
        type preference-level;
        default "prefer";
        description "Section 5.2.2 of draft-ietf-taps-interface-03";
    }

    leaf per-message-reliability {
        type preference-level;
        default "ignore";
        description "Section 5.2.3 of draft-ietf-taps-interface-03";
    }
}
leaf preserve-order {
  type preference-level;
  default "require";
  description "Section 5.2.4 of draft-ietf-taps-interface-03";
}

leaf zero-rtt-msg {
  type preference-level;
  default "prefer";
  description "Section 5.2.5 of draft-ietf-taps-interface-03";
}

leaf multistreaming {
  type preference-level;
  default "prefer";
  description "Section 5.2.6 of draft-ietf-taps-interface-03";
}

leaf per-msg-checksum-len-send {
  type preference-level;
  default "ignore";
  description "Section 5.2.7 of draft-ietf-taps-interface-03";
}

leaf per-msg-checksum-len-recv {
  type preference-level;
  default "ignore";
  description "Section 5.2.8 of draft-ietf-taps-interface-03";
}

leaf congestion-control {
  type preference-level;
  default "require";
  description "Section 5.2.9 of draft-ietf-taps-interface-03";
}

list interface {
  key "preference value";
  leaf preference {
    type preference-level;
    description "preference for this interface or interface type";
  }
  leaf value {
    type union {
      type identityref {
        base "ianaift:iana-interface-type";
      }
      type if:interface-ref {
        base "if:interface-ref";
      }
    }
  }
}
require-instance false;
}
}
description "name or type of interface constraint";
reference "RFC 7224 Section 2";
}

description "Section 5.2.10 of draft-ietf-taps-interface-03";
}
list pvd {
  key "preference value";
  leaf preference {
    type preference-level;
    description "preference for this pvd";
  }
  leaf value {
    type identityref {
      base "provisioning-domain";
    }
    description "the provisioning domain constraint";
  }
  description "Section 5.2.11 of draft-ietf-taps-interface-03";
}
leaf multipath {
  type preference-level;
  default "prefer";
  description "Section 5.2.12 of draft-ietf-taps-interface-03";
}
leaf direction {
  type enumeration {
    enum bidirectional {
      description "Bidirectional connection";
    }
    enum unidirection-send {
      description "Unidirectional sending connection";
    }
    enum unidirection-receive {
      description "Unidirectional receiving connection";
    }
  }
  default "bidirectional";
  description "Section 5.2.13 of draft-ietf-taps-interface-03";
}
leaf retransmit-notify {
  type preference-level;
  default "ignore";
  description "Section 5.2.14 of draft-ietf-taps-interface-03";
}

leaf soft-error-notify {
  type preference-level;
  default "ignore";
  description "Section 5.2.15 of draft-ietf-taps-interface-03";
}

container security {
  description "Security properties for the connection";
  list credentials {
    key "identity algorithm";
    uses security-credentials;
    description "security credentials";
  }
  leaf session-cache-capacity {
    type uint32;
    description "Max number of cache elements";
  }
  leaf session-cache-lifetime {
    type uint32;
    description "Number of seconds of session cache lifetime";
  }
}

Figure 5: TAPS Interface YANG model

5. 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.)

6. IANA Considerations

IANA is requested to add the YANG model in Section 4 to the yang-parameters registry.

+----------+------------------------------------------+
| Field    | Value                                    |
+----------+------------------------------------------+
| Name     | ietf-taps-api                            |
| Prefix   | taps                                     |
| Reference| [TBD: this document]                      |

7. Normative References

[I-D.ietf-taps-arch]

[I-D.ietf-taps-interface]


Appendix A.  Future Work

TBD if adopted:

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

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

o 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