

Babel Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 30, 2020

M. Jethanandani
Kloud Services
B. Stark
AT&T
June 28, 2020

YANG Data Model for Babel
draft-ietf-babel-yang-model-06

Abstract

This document defines a data model for the Babel routing protocol. The data model is defined using the YANG data modeling language.

Requirements Language

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.

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 30, 2020.

Copyright Notice

Copyright (c) 2020 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.	Note to RFC Editor	2
1.2.	Tree Diagram Annotations	3
2.	Babel Module	3
2.1.	Information Model	3
2.2.	Tree Diagram	3
2.3.	YANG Module	4
3.	IANA Considerations	29
3.1.	URI Registrations	29
3.2.	YANG Module Name Registration	29
4.	Security Considerations	29
5.	Acknowledgements	30
6.	References	30
6.1.	Normative References	30
6.2.	Informative References	31
Appendix A.	An Appendix	32
A.1.	Statistics Gathering Enabled	32
A.2.	Automatic Detection of Properties	34
A.3.	Override Default Properties	35
A.4.	Configuring other Properties	36
	Authors' Addresses	38

[1.](#) Introduction

This document defines a data model for the Babel routing protocol [[I-D.ietf-babel-rfc6126bis](#)]. The data model is defined using YANG 1.1 [[RFC7950](#)] data modeling language and is Network Management Datastore Architecture (NDMA) [[RFC8342](#)] compatible. It is based on the Babel Information Model [[I-D.ietf-babel-information-model](#)].

[1.1.](#) Note to RFC Editor

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements and remove this

note before publication.

- o "XXXX" --> the assigned RFC value for this draft both in this draft and in the YANG models under the revision statement.

- o "ZZZZ" --> the assigned RFC value for Babel Information Model [[I-D.ietf-babel-information-model](#)]
- o Revision date in model, in the format 2020-06-28 needs to get updated with the date the draft gets approved. The date also needs to get reflected on the line with <CODE BEGINS>.

[1.2.](#) Tree Diagram Annotations

For a reference to the annotations used in tree diagrams included in this draft, please see YANG Tree Diagrams [[RFC8340](#)].

[2.](#) Babel Module

This document defines a YANG 1.1 [[RFC7950](#)] data model for the configuration and management of Babel. The YANG module is based on the Babel Information Model [[I-D.ietf-babel-information-model](#)].

[2.1.](#) Information Model

There are a few things that should be noted between the Babel Information Model and this data module. The information model mandates the definition of some of the attributes, e.g. `babel-implementation-version` or the `babel-self-router-id`. These attributes are marked as read-only objects in the information module as well as in this data module. However, there is no way in the data module to mandate that a read-only attribute be present. It is up to the implementation of this data module to make sure that the attributes that are marked read-only and are mandatory are indeed present.

[2.2.](#) Tree Diagram

The following diagram illustrates a top level hierarchy of the model. In addition to information like the version number implemented by this device, the model contains subtrees on constants, interfaces, routes and security.

```
module: ietf-babel
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol:
      +--rw babel!
        +--ro version?      string
        +--rw enable        boolean
        +--ro router-id?    binary
        +--ro seqno?        uint16
        +--rw stats-enable? boolean
        +--rw constants
          |   ...
        +--rw interfaces* [reference]
          |   ...
        +--rw mac* [name]
          |   ...
        +--rw dtls* [name]
          |   ...
        +--ro routes* [prefix]
          ...
```

The interfaces subtree describes attributes such as interface object that is being referenced, the type of link as enumerated by metric-algorithm and split-horizon and whether the interface is enabled or not.

The constants subtree describes the UDP port used for sending and receiving Babel messages, and the multicast group used to send and receive announcements on IPv6.

The routes subtree describes objects such as the prefix for which the route is advertised, a reference to the neighboring route, and next-hop address.

Finally, for security two subtree are defined to contain MAC keys and DTLS certificates. The mac subtree contains keys used with the MAC security mechanism. The boolean flag default-apply indicates whether the set of MAC keys is automatically applied to new interfaces. The dtls subtree contains certificates used with DTLS security mechanism. Similar to the MAC mechanism, the boolean flag default-apply indicates whether the set of DTLS certificates is automatically applied to new interfaces.

[2.3.](#) YANG Module

This YANG module augments the YANG Routing Management [[RFC8349](#)] module to provide a common framework for all routing subsystems. By augmenting the module it provides a common building block for routes, and Routing Information Bases (RIBs). It also has a reference to an

interface defined by A YANG Data Model for Interface Management [[RFC8343](#)].

A router running Babel routing protocol can determine the parameters it needs to use for an interface based on the interface name. For example, it can detect that eth0 is a wired interface, and that wlan0 is a wireless interface. This is not true for a tunnel interface, where the link parameters need to be configured explicitly.

For a wired interface, it will assume '2-out-of-3' 'metric-algorithm', and 'split-horizon' set to true. On other hand, for a wireless interface it will assume 'etx' 'metric-algorithm', and 'split-horizon' set to false. However, if the wired link is connected to a wireless radio, the values can be overridden by setting 'metric-algorithm' to 'etx', and 'split-horizon' to false. Similarly, an interface that is a metered 3G link, and used for fallback connectivity needs much higher default time constants, e.g. 'mcast-hello-interval', and 'update-interval', in order to avoid carrying control traffic as much as possible.

In addition to the modules used above, this module imports definitions from Common YANG Data Types [[RFC6991](#)], and references

HMAC: Keyed-Hashing for Message Authentication [[RFC2104](#)], Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 [[RFC4868](#)], Datagram Transport Layer Security Version 1.2 [[RFC6347](#)], The Blake2 Cryptographic Hash and Message Authentication Code (MAC) [[RFC7693](#)], Babel Information Model [[I-D.ietf-babel-information-model](#)], and The Babel Routing Protocol [[I-D.ietf-babel-rfc6126bis](#)].

<CODE BEGINS> file "ietf-babel@2020-06-28.yang"

```
module ietf-babel {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-babel";
  prefix babel;

  import ietf-yang-types {
    prefix yt;
    reference
      "RFC 6991: Common YANG Data Types.";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types.";
  }
  import ietf-interfaces {
    prefix if;
```

```
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }
  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: YANG Routing Management";
  }

  organization
    "IETF Babel routing protocol Working Group";

  contact
    "WG Web: http://tools.ietf.org/wg/babel/
     WG List: babel@ietf.org
```

Editor: Mahesh Jethanandani
mjethanandani@gmail.com
Editor: Barbara Stark
bs7652@att.com";

description

"This YANG module defines a model for the Babel routing protocol.

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) 2020 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 (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX (<https://www.rfc-editor.org/info/rfcXXXX>); see the RFC itself for full legal notices.";

revision 2020-06-28 {
description
"Initial version.";

reference
"RFC XXXX: Babel YANG Data Model.";
}

/*
* Features
*/
feature two-out-of-three-supported {

```
description
  "This implementation supports two-out-of-three metric
  comp algorithm.";
}

feature etx-supported {
  description
    "This implementation supports Expected Transmission Count
    (ETX) metric comp algorithm.";
}

feature mac-supported {
  description
    "This implementation supports MAC based security.";
}

feature dtls-supported {
  description
    "This implementation supports DTLS based security.";
}

feature hmac-sha256-supported {
  description
    "This implementation supports hmac-sha256 MAC algorithm.";
}

feature blake2s-supported {
  description
    "This implementation supports blake2 MAC algorithm.";
}

feature x-509-supported {
  description
    "This implementation supports x-509 certificate type.";
}

feature raw-public-key-supported {
  description
    "This implementation supports raw-public-key certificate type.";
}
```



```

* Identities
*/
identity metric-comp-algorithms {
    description
        "Base identity from which all Babel metric comp algorithms
        are derived.";
}

identity two-out-of-three {
    if-feature two-out-of-three-supported;
    base "metric-comp-algorithms";
    description
        "2-out-of-3 algorithm.";
}

identity etx {
    if-feature etx-supported;
    base "metric-comp-algorithms";
    description
        "Expected Transmission Count.";
}

/*
* Babel MAC algorithms identities.
*/
identity mac-algorithms {
    description
        "Base identity for all Babel MAC algorithms.";
}

identity hmac-sha256 {
    if-feature mac-supported;
    if-feature hmac-sha256-supported;
    base mac-algorithms;
    description
        "HMAC-SHA256 algorithm supported.";
    reference
        "RFC 4868: Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512
        with IPsec.";
}

identity blake2s {
    if-feature mac-supported;
    if-feature blake2s-supported;
    base mac-algorithms;
    description
        "BLAKE2s algorithm supported.";
}

```

```
    reference
      "RFC 7693: The BLAKE2 Cryptographic Hash and Message
      Authentication Code (MAC).";
  }

  /*
   * Babel Cert Types
   */
  identity dtls-cert-types {
    description
      "Base identity for Babel DTLS certificate types.";
  }

  identity x-509 {
    if-feature dtls-supported;
    if-feature x-509-supported;
    base dtls-cert-types;
    description
      "X.509 certificate type.";
  }

  identity raw-public-key {
    if-feature dtls-supported;
    if-feature raw-public-key-supported;
    base dtls-cert-types;
    description
      "Raw Public Key type.";
  }

  /*
   * Babel routing protocol identity.
   */
  identity babel {
    base "rt:routing-protocol";
    description
      "Babel routing protocol";
  }

  /*
   * Groupings
   */
  grouping routes {
    list routes {
      key "prefix";
      config false;
    }
  }
}
```

```
leaf prefix {
  type inet:ip-prefix;
```

```
  description
    "Prefix (expressed in ip-address/prefix-length format) for
     which this route is advertised.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.6.";
}
```

```
leaf router-id {
  type binary;
  description
    "router-id of the source router for which this route is
     advertised.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.6.";
}
```

```
leaf neighbor {
  type leafref {
    path "/rt:routing/rt:control-plane-protocols/" +
         "rt:control-plane-protocol/babel/interfaces/" +
         "neighbor-objects/neighbor-address";
  }
  description
    "Reference to the neighbor-objects entry for the neighbor
     that advertised this route.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.6.";
}
```

```
leaf received-metric {
  type uint16;
  description
    "The metric with which this route was advertised by the
     neighbor, or maximum value (infinity) to indicate the
     route was recently retracted and is temporarily
     unreachable. This metric will be 0 (zero) if the route
     was not received from a neighbor but was generated
     through other means. At least one of
     calculated-metric or received-metric MUST be non-NULL.";
```

```

reference
  "RFC ZZZZ: Babel Information Model, Section 3.6,
  draft-ietf-babel-rfc6126bis: The Babel Routing Protocol,
  Section 3.5.5.";
}

leaf calculated-metric {
  type uint16;
  description

```

```

  "A calculated metric for this route. How the metric is
  calculated is implementation-specific. Maximum value
  (infinity) indicates the route was recently retracted
  and is temporarily unreachable. At least one of
  calculated-metric or received-metric MUST be non-NULL.";
reference
  "RFC ZZZZ: Babel Information Model, Section 3.6,
  draft-ietf-babel-rfc6126bis: The Babel Routing Protocol,
  Section 3.5.5.";
}

leaf seqno {
  type uint16;
  description
    "The sequence number with which this route was advertised.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.6.";
}

leaf next-hop {
  type inet:ip-address;
  description
    "The next-hop address of this route. This will be empty if
    this route has no next-hop address.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.6.";
}

leaf feasible {
  type boolean;
  description
    "A boolean flag indicating whether this route is feasible.";

```

```

reference
  "RFC ZZZZ: Babel Information Model, Section 3.6,
  draft-ietf-babel-rfc6126bis, The Babel Routing Protocol,
  Section 3.5.1.";
}

leaf selected {
  type boolean;
  description
    "A boolean flag indicating whether this route is selected,
    i.e., whether it is currently being used for forwarding and
    is being advertised.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.6.";
}
description

```

```

  "A set of babel-route-obj objects. Includes received and
  routes routes.";
reference
  "RFC ZZZZ: Babel Information Model, Section 3.1.";
}
description
  "Common grouping for routing used in RIB.";
}

/*
 * Data model
 */

augment "/rt:routing/rt:control-plane-protocols/" +
  "rt:control-plane-protocol" {
  when "derived-from-or-self(rt:type, 'babel')" {
    description
      "Augmentation is valid only when the instance of routing type
      is of type 'babel'.";
  }
  description
    "Augment the routing module to support a common structure
    between routing protocols.";
  reference
    "YANG Routing Management, RFC 8349, Lhotka & Lindem, March

```

```

2018.";

container babel {
  presence "A Babel container.";
  description
    "Babel Information Objects.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.";

  leaf version {
    type string;
    config false;
    description
      "The name and version of this implementation of the Babel
        protocol.";
    reference
      "RFC ZZZZ: Babel Information Model, Section 3.1.";
  }

  leaf enable {
    type boolean;
    mandatory true;
    description

```

```

    "When written, it configures whether the protocol should be
      enabled. A read from the <running> or <intended> datastore
      therefore indicates the configured administrative value of
      whether the protocol is enabled or not.

      A read from the <operational> datastore indicates whether
      the protocol is actually running or not, i.e. it indicates
      the operational state of the protocol.";
    reference
      "RFC ZZZZ: Babel Information Model, Section 3.1.";
  }

  leaf router-id {
    type binary;
    config false;
    description
      "Every Babel speaker is assigned a router-id, which is an
        arbitrary string of 8 octets that is assumed to be unique

```

```

        across the routing domain";
reference
    "RFC ZZZZ: Babel Information Model, Section 3.1,
    draft-ietf-babel-rfc6126bis: The Babel Routing Protocol,
    Section 3.";
}

leaf seqno {
    type uint16;
    config false;
    description
        "Sequence number included in route updates for routes
        originated by this node.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.1.";
}

leaf stats-enable {
    type boolean;
    description
        "Indicates whether statistics collection is enabled (true)
        or disabled (false) on all interfaces.";
}

container constants {
    description
        "Babel Constants object.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.1.";
}

```

```

leaf udp-port {
    type inet:port-number;
    default "6696";
    description
        "UDP port for sending and receiving Babel messages. The
        default port is 6696.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.2.";
}

leaf mcast-group {

```

```

    type inet:ip-address;
    default "ff02::1:6";
    description
        "Multicast group for sending and receiving multicast
        announcements on IPv6.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.2.";
}
}

list interfaces {
    key "reference";

    description
        "A set of Babel Interface objects.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.3.";

    leaf reference {
        type if:interface-ref;
        description
            "References the name of the interface over which Babel
            packets are sent and received.";
        reference
            "RFC ZZZZ: Babel Information Model, Section 3.3.";
    }

    leaf enable {
        type boolean;
        default "true";
        description
            "If true, babel sends and receives messages on this
            interface. If false, babel messages received on this
            interface are ignored and none are sent.";
        reference
            "RFC ZZZZ: Babel Information Model, Section 3.3.";
    }
}

```

```

leaf metric-algorithm {
    type identityref {
        base metric-comp-algorithms;
    }
}

```



```

    mandatory true;
    description
        "Indicates the metric computation algorithm used on this
        interface. The value MUST be one of those identities
        based on 'metric-comp-algorithms'.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.X.";
}

leaf split-horizon {
    type boolean;
    description
        "Indicates whether or not the split horizon optimization
        is used when calculating metrics on this interface.
        A value of true indicates split horizon optimization
        is used.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.X.";
}

leaf mcast-hello-seqno {
    type uint16;
    config false;
    description
        "The current sequence number in use for multicast hellos
        sent on this interface.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.3.";
}

leaf mcast-hello-interval {
    type uint16;
    units centiseconds;
    description
        "The current multicast hello interval in use for hellos
        sent on this interface.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.3.";
}

leaf update-interval {
    type uint16;
    units centiseconds;
    description

```

```
        "The current update interval in use for this interface.
        Units are centiseconds.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.3.";
}

leaf mac-enable {
    type boolean;
    description
        "Indicates whether the MAC security mechanism is enabled
        (true) or disabled (false).";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.3.";
}

leaf-list mac-key-sets {
    type leafref {
        path "../..../mac/name";
    }
    description
        "List of references to the mac entries that apply
        to this interface. When an interface instance is
        created, all mac instances with default-apply 'true'
        will be included in this list.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.3.";
}

leaf mac-verify {
    type boolean;
    description
        "A Boolean flag indicating whether MAC hashes in
        incoming Babel packets are required to be present and
        are verified. If this parameter is 'true', incoming
        packets are required to have a valid MAC hash.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.3.";
}

leaf dtls-enable {
    type boolean;
    description
        "Indicates whether the DTLS security mechanism is enabled
        (true) or disabled (false).";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.3.";
}
```

```
leaf-list dtls-certs {
  type leafref {
    path "../..//dtls/name";
  }
  description
    "List of references to the dtls entries that apply to
    this interface. When an interface instance
    is created, all dtls instances with default-apply
    'true' will be included in this list.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.3.";
}

leaf dtls-cached-info {
  type boolean;
  description
    "Indicates whether the cached_info extension is included
    in ClientHello and ServerHello packets. The extension
    is included if the value is 'true'.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.3.";
}

leaf-list dtls-cert-prefer {
  type leafref {
    path "../..//dtls/certs/type";
  }
  ordered-by user;
  description
    "List of supported certificate types, in order of
    preference. The values MUST be among those listed in
    dtls-cert-types. This list is used to populate the
    server_certificate_type extension in a Client Hello.
    Values that are present in at least one instance in the
    certs object under dtls of a referenced dtls instance
    and that have a non-empty private-key will be used to
    populate the client_certificate_type extension in a
    Client Hello.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.3.";
}
```

```
}

leaf packet-log-enable {
  type boolean;
  description
    "If true, logging of babel packets received on this
    interface is enabled; if false, babel packets are not
    logged.";
```

```
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.3.";
}

leaf packet-log {
  type inet:uri;
  config false;
  description
    "A reference or url link to a file that contains a
    timestamped log of packets received and sent on
    udp-port on this interface. The [libpcap] file
    format with .pcap file extension SHOULD be supported for
    packet log files. Logging is enabled / disabled by
    packet-log-enable.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.3.";
}

container stats {
  config false;

  description
    "Statistics collection object for this interface.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.3.";

  leaf sent-mcast-hello {
    type yt:counter32;
    description
      "A count of the number of multicast Hello packets sent
      on this interface.";
    reference
      "RFC ZZZZ: Babel Information Model, Section 3.4.";
```

```
}

leaf sent-mcast-update {
  type yt:counter32;
  description
    "A count of the number of multicast update packets sent
    on this interface.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.4.";
}

leaf sent-ucast-hello {
  type yt:counter32;
  description
    "A count of the number of unicast Hello packets sent
```

```
    to this neighbor.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.6.";
}

leaf sent-ucast-update {
  type yt:counter32;
  description
    "A count of the number of unicast update packets sent
    to this neighbor.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.6.";
}

leaf sent-ihu {
  type yt:counter32;
  description
    "A count of the number of IHU packets sent to this
    neighbor.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.6.";
}

leaf received-packets {
  type yt:counter32;
  description
```

```

    "A count of the number of Babel packets received on
      this interface.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.4.";
}
action reset {
  description
    "The information model [RFC ZZZZ] defines reset
      action as a system-wide reset of Babel statistics.
      In YANG the reset action is associated with the
      container where the action is defined. In this case
      the action is associated with the stats container
      inside an interface. The action will therefore
      reset statistics at an interface level.

      Implementations that want to support a system-wide
      reset of Babel statistics need to call this action
      for every instance of the interface.";

  input {
    leaf reset-at {
      type yt:date-and-time;

```

```

      description
        "The time when the reset was issued.";
    }
  }
  output {
    leaf reset-finished-at {
      type yt:date-and-time;
      description
        "The time when the reset finished.";
    }
  }
}
}

list neighbor-objects {
  key "neighbor-address";
  config false;

  description

```

```

    "A set of Babel Neighbor Object.";
reference
    "RFC ZZZZ: Babel Information Model, Section 3.5.";

leaf neighbor-address {
    type inet:ip-address;
    description
        "IPv4 or v6 address the neighbor sends packets from.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.5.";
}

leaf hello-mcast-history {
    type string;
    description
        "The multicast Hello history of whether or not the
        multicast Hello packets prior to exp-mcast-
        hello-seqno were received, with a '1' for the most
        recent Hello placed in the most significant bit and
        prior Hellos shifted right (with '0' bits placed
        between prior Hellos and most recent Hello for any
        not-received Hellos); represented as a string using
        utf-8 encoded hex digits where a '1' bit = Hello
        received and a '0' bit = Hello not received.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.5.";
}

leaf hello-ucast-history {

```

```

type string;
description
    "The unicast Hello history of whether or not the
    unicast Hello packets prior to exp-ucast-hello-seqno
    were received, with a '1' for the most
    recent Hello placed in the most significant bit and
    prior Hellos shifted right (with '0' bits placed
    between prior Hellos and most recent Hello for any
    not-received Hellos); represented as a string using
    utf-8 encoded hex digits where a '1' bit = Hello
    received and a '0' bit = Hello not received.";
reference

```

```

    "RFC ZZZZ: Babel Information Model, Section 3.5.";
}

leaf txcost {
    type int32;
    default "0";
    description
        "Transmission cost value from the last IHU packet
         received from this neighbor, or maximum value
         (infinity) to indicate the IHU hold timer for this
         neighbor has expired description.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.5.";
}

leaf exp-mcast-hello-seqno {
    type uint16;
    default "0";
    description
        "Expected multicast Hello sequence number of next Hello
         to be received from this neighbor; if multicast Hello
         packets are not expected, or processing of multicast
         packets is not enabled, this MUST be NULL.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.5.";
}

leaf exp-ucast-hello-seqno {
    type uint16;
    default "0";
    description
        "Expected unicast Hello sequence number of next Hello to
         be received from this neighbor; if unicast Hello
         packets are not expected, or processing of unicast
         packets is not enabled, this MUST be NULL.";
    reference

```

```

    "RFC ZZZZ: Babel Information Model, Section 3.5.";
}

leaf ucast-hello-seqno {
    type uint16;

```



```

description
  "Expected unicast Hello sequence number of next Hello
   to be received from this neighbor. If unicast Hello
   packets are not expected, or processing of unicast
   packets is not enabled, this MUST be 0.";
reference
  "RFC ZZZZ: Babel Information Model, Section 3.5.";
}

leaf ucast-hello-interval {
  type uint16;
  units centiseconds;
  description
    "The current interval in use for unicast hellos sent to
     this neighbor. Units are centiseconds.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.5.";
}

leaf rxcost {
  type int32;
  description
    "Reception cost calculated for this neighbor. This value
     is usually derived from the Hello history, which may be
     combined with other data, such as statistics maintained
     by the link layer. The rxcost is sent to a neighbor in
     each IHU.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.5.";
}

leaf cost {
  type int32;
  description
    "Link cost is computed from the values maintained in
     the neighbor table. The statistics kept in the neighbor
     table about the reception of Hellos, and the txcost
     computed from received IHU packets.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.5.";
}
}
}

```

```

list mac {
    key "name";

    description
        "A mac object. If this object is implemented, it
        provides access to parameters related to the MAC security
        mechanism.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.7.";

    leaf name {
        type string;
        description
            "A string that uniquely identifies the mac object.";
    }

    leaf default-apply {
        type boolean;
        description
            "A Boolean flag indicating whether this mac
            instance is applied to all new interfaces, by default.
            If 'true', this instance is applied to new
            interfaces instances at the time they are created,
            by including it in the mac-key-sets list under
            interfaces. If 'false', this instance is not applied
            to new interfaces instances when they are created.";
        reference
            "RFC ZZZZ: Babel Information Model, Section 3.7.";
    }

    list keys {
        key "name";
        min-elements "1";

        description
            "A set of keys objects.";
        reference
            "RFC ZZZZ: Babel Information Model, Section 3.8.";

        leaf name {
            type string;
            mandatory true;
            description
                "A unique name for this MAC key that can be used to
                identify the key in this object instance, since the key
                value is not allowed to be read. This value can only be
                provided when this instance is created, and is not
                subsequently writable.";
        }
    }
}

```

Internet-Draft

Babel YANG model

June 2020

```
reference
  "RFC ZZZZ: Babel Information Model, Section 3.8.";
}

leaf use-sign {
  type boolean;
  mandatory true;
  description
    "Indicates whether this key value is used to sign sent
    Babel packets. Sent packets are signed using this key
    if the value is 'true'. If the value is 'false', this
    key is not used to sign sent Babel packets.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.8.";
}

leaf use-verify {
  type boolean;
  mandatory true;
  description
    "Indicates whether this key value is used to verify
    incoming Babel packets. This key is used to verify
    incoming packets if the value is 'true'. If the value
    is 'false', no MAC is computed from this key for
    comparing an incoming packet.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.8.";
}

leaf value {
  type binary;
  mandatory true;
  description
    "The value of the MAC key. An implementation MUST NOT
    allow this parameter to be read. This can be done by
    always providing an empty string, or through
    permissions, or other means. This value MUST be
    provided when this instance is created, and is not
    subsequently writable.

    This value is of a length suitable for the associated
    algorithm. If the algorithm is based on
```

the HMAC construction [[RFC2104](#)], the length MUST be between 0 and the block size of the underlying hash inclusive (where 'HMAC-SHA256' block size is 64 bytes as described in [[RFC4868](#)]). If the algorithm is 'BLAKE2s', the length MUST be between 0 and 32 bytes inclusive, as described in [[RFC7693](#)].";

```
reference
  "RFC ZZZZ: Babel Information Model, Section 3.8,
  RFC 2104: HMAC: Keyed-Hashing for Message
  Authentication
  RFC 4868: Using HMAC-SHA-256, HMAC-SHA-384, and
  HMAC-SHA-512 with IPsec,
  RFC 7693: The BLAKE2 Cryptographic Hash and Message
  Authentication Code (MAC).";
}

leaf algorithm {
  type identityref {
    base mac-algorithms;
  }
  description
    "The name of the MAC algorithm used with this key. The
    value MUST be the same as one of the enumerations
    listed in the mac-algorithms parameter.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.8.";
}

action test {
  description
    "An operation that allows the MAC key and hash
    algorithm to be tested to see if they produce an
    expected outcome. Input to this operation is a
    binary string. The implementation is expected to
    create a hash of this string using the value and
    the algorithm. The output of this operation is
    the resulting hash, as a binary string.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.8.";

  input {
```

```

leaf test-string {
  type binary;
  mandatory true;
  description
    "Input to this operation is a binary string.
    The implementation is expected to create
    a hash of this string using the value and
    the algorithm.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.8.";
}
}
output {

```

```

leaf resulting-hash {
  type binary;
  mandatory true;
  description
    "The output of this operation is
    the resulting hash, as a binary string.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.8.";
}
}
}
}
}

list dtls {
  key "name";

  description
    "A dtls object. If this object is implemented,
    it provides access to parameters related to the DTLS
    security mechanism.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.9";

  leaf name {
    type string;
    description
      "A string that uniquely identifies a dtls object.";

```

```

}

leaf default-apply {
  type boolean;
  mandatory true;
  description
    "A Boolean flag indicating whether this dtls
    instance is applied to all new interfaces, by default. If
    'true', this instance is applied to new interfaces
    instances at the time they are created, by including it
    in the dtls-certs list under interfaces. If 'false',
    this instance is not applied to new interfaces
    instances when they are created.";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.9.";
}

list certs {
  key "name";
  min-elements "1";

```

```

description
  "A set of cert objects. This contains
  both certificates for this implementation to present
  for authentication, and to accept from others.
  Certificates with a non-empty private-key
  can be presented by this implementation for
  authentication.";
reference
  "RFC ZZZZ: Babel Information Model, Section 3.10.";

leaf name {
  type string;
  description
    "A unique name for this DTLS certificate that can be
    used to identify the certificate in this object
    instance, since the value is too long to be useful
    for identification. This value MUST NOT be empty
    and can only be provided when this instance is created
    (i.e., it is not subsequently writable).";
  reference
    "RFC ZZZZ: Babel Information Model, Section 3.10.";

```

```

}

leaf value {
    type string;
    mandatory true;
    description
        "The DTLS certificate in PEM format [RFC7468]. This
        value can only be provided when this instance is
        created, and is not subsequently writable.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.10.";
}

leaf type {
    type identityref {
        base dtls-cert-types;
    }
    mandatory true;
    description
        "The name of the certificate type of this object
        instance. The value MUST be the same as one of the
        enumerations listed in the dtls-cert-types
        parameter. This value can only be provided when this
        instance is created, and is not subsequently writable.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.10.";
}

```

```

leaf private-key {
    type binary;
    mandatory true;
    description
        "The value of the private key. If this is non-empty,
        this certificate can be used by this implementation to
        provide a certificate during DTLS handshaking. An
        implementation MUST NOT allow this parameter to be
        read. This can be done by always providing an empty
        string, or through permissions, or other means. This
        value can only be provided when this instance is
        created, and is not subsequently writable.";
    reference
        "RFC ZZZZ: Babel Information Model, Section 3.10.";
}

```

```
    }

    action test {
      input {
        leaf test-string {
          type binary;
          mandatory true;
          description
            "The test string on which this test has to be
            performed.";
        }
      }
      output {
        leaf resulting-hash {
          type binary;
          mandatory true;
          description
            "The output of this operation is a binary string,
            and is the resulting hash computed using the
            certificate public key, and the SHA-256
            hash algorithm.";
        }
      }
    }
  }
}

uses routes;
}
}
}

<CODE ENDS>
```

[3.](#) IANA Considerations

This document registers one URIs and one YANG module.

[3.1.](#) URI Registrations

URI: urn:ietf:params:xml:ns:yang:ietf-babel

3.2. YANG Module Name Registration

This document registers one YANG module in the YANG Module Names registry YANG [[RFC6020](#)].

Name: ietf-babel
Namespace: urn:ietf:params:xml:ns:yang:ietf-babel
prefix: babel
reference: RFC XXXX

4. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocol such as NETCONF [[RFC6241](#)] or RESTCONF [[RFC8040](#)]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [[RFC6242](#)]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [[RFC8446](#)].

The NETCONF Access Control Model (NACM [[RFC8341](#)]) provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in the YANG module which are writable/created/deleted (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 from a config true perspective:

babel: This container includes an "enable" parameter that can be used to enable or disable use of Babel on a router

babel/constants: This container includes configuration parameters that can prevent reachability if misconfigured.

babel/interfaces: This leaf-list has configuration parameters that can enable/disable security mechanisms and change performance characteristics of the Babel protocol.

babel/hmac and babel/dtls: These contain security credentials that influence whether packets are trusted.

Some of the readable data or config false 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 from a config false perspective:

babel: Access to the information in the various nodes can disclose the network topology. Additionally, the routes used by a network device may be used to mount a subsequent attack on traffic traversing the network device.

babel/hmac and babel/dtls: These contain security credentials, include private credentials of the router.

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability from a RPC operation perspective:

babel/hmac/hmac/keys/test and babel/dtls/certs/test: These can be used in a brute force attack to identify the credentials being used to secure the Babel protocol.

5. Acknowledgements

Juliusz Chroboczek provided most of the example configurations for babel that are shown in the Appendix.

6. References

6.1. Normative References

[I-D.ietf-babel-rfc6126bis]

Chroboczek, J. and D. Schinazi, "The Babel Routing Protocol", [draft-ietf-babel-rfc6126bis-17](#) (work in progress), February 2020.

[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>>.

Internet-Draft

Babel YANG model

June 2020

- [RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", [RFC 4868](#), DOI 10.17487/RFC4868, May 2007, <<https://www.rfc-editor.org/info/rfc4868>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", [RFC 6347](#), DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [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>>.
- [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>>.

[6.2.](#) Informative References

- [I-D.ietf-babel-information-model]
Stark, B. and M. Jethanandani, "Babel Information Model", [draft-ietf-babel-information-model-10](#) (work in progress), October 2019.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", [RFC 2104](#), DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.
- [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>>.

- [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>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", [RFC 6242](#), DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC7693] Saarinen, M-J., Ed. and J-P. Aumasson, "The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC)", [RFC 7693](#), DOI 10.17487/RFC7693, November 2015, <<https://www.rfc-editor.org/info/rfc7693>>.
- [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>>.
- [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>>.
- [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>>.

[Appendix A](#). An Appendix

This section is devoted to examples that demonstrate how Babel can be configured.

[A.1](#). Statistics Gathering Enabled

In this example, interface eth0 is being configured for routing protocol Babel, and statistics gathering is enabled. For security, HMAC-SHA256 is supported. Every sent Babel packets is signed with the key value provided, and every received Babel packet is verified with the same key value.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
             xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
      <enabled>true</enabled>
    </interface>
  </interfaces>
  <routing
    xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type
          xmlns:babel=
            "urn:ietf:params:xml:ns:yang:ietf-babel">babel:babel
        </type>
        <name>name:babel</name>
        <babel
          xmlns="urn:ietf:params:xml:ns:yang:ietf-babel">
          <enable>true</enable>
          <stats-enable>true</stats-enable>
          <interfaces>
            <reference>eth0</reference>
            <metric-algorithm>two-out-of-three</metric-algorithm>
            <split-horizon>true</split-horizon>
          </interfaces>
          <mac>
```

```

    <name>hmac-sha256</name>
    <keys>
      <name>hmac-sha256-keys</name>
      <use-sign>true</use-sign>
      <use-verify>true</use-verify>
      <value>base64encodedvalue==</value>
      <algorithm>hmac-sha256</algorithm>
    </keys>
  </mac>
</babel>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</config>

```

[A.2.](#) Automatic Detection of Properties

<!-- In this example, babeld is configured on two interfaces

```

interface eth0
interface wlan0

```

This says to run Babel on interfaces eth0 and wlan0. Babeld will automatically detect that eth0 is wired and wlan0 is wireless, and will configure the right parameters automatically.

-->

```

<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
             xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
      <enabled>true</enabled>
    </interface>
    <interface>

```

```

    <name>wlan0</name>
    <type>ianaift:ieee80211</type>
    <enabled>true</enabled>
  </interface>
</interfaces>
<routing
  xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
  <control-plane-protocols>
    <control-plane-protocol>
      <type
        xmlns:babel=
          "urn:ietf:params:xml:ns:yang:ietf-babel">babel:babel
      </type>
      <name>name:babel</name>
      <babel
        xmlns="urn:ietf:params:xml:ns:yang:ietf-babel">
        <enable>true</enable>
        <interfaces>
          <reference>eth0</reference>
          <enable>true</enable>
          <metric-algorithm>two-out-of-three</metric-algorithm>
          <split-horizon>true</split-horizon>
        </interfaces>
        <interfaces>
          <reference>wlan0</reference>
          <enable>true</enable>

```

```

    <metric-algorithm>etx</metric-algorithm>
    <split-horizon>>false</split-horizon>
  </interfaces>
</babel>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</config>

```

[A.3.](#) Override Default Properties

<!-- In this example, babeld is configured on three interfaces

```

interface eth0
interface eth1 type wireless

```

```
interface tun0 type tunnel
```

Here, interface eth1 is an Ethernet bridged to a wireless radio, so babeld's autodetection fails, and the interface type needs to be configured manually. Tunnels are not detected automatically, so this needs to be specified.

This is equivalent to the following:

```
interface eth0 metric-algorithm 2-out-of-3 split-horizon true
interface eth1 metric-algorithm etx split-horizon false
interface tun0 metric-algorithm 2-out-of-3 split-horizon true
-->
```

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
             xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
      <enabled>true</enabled>
    </interface>
    <interface>
      <name>eth1</name>
      <type>ianaift:ethernetCsmacd</type>
      <enabled>true</enabled>
    </interface>
    <interface>
      <name>tun0</name>
      <type>ianaift:tunnel</type>
      <enabled>true</enabled>
    </interface>
  </interfaces>

```

```
</interfaces>
<routing
  xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
  <control-plane-protocols>
    <control-plane-protocol>
      <type
        xmlns:babel=
          "urn:ietf:params:xml:ns:yang:ietf-babel">babel:babel

```



```

</type>
<name>name:babel</name>
<babel
  xmlns="urn:ietf:params:xml:ns:yang:ietf-babel">
  <enable>true</enable>
  <interfaces>
    <reference>eth0</reference>
    <enable>true</enable>
    <metric-algorithm>two-out-of-three</metric-algorithm>
    <split-horizon>true</split-horizon>
  </interfaces>
  <interfaces>
    <reference>eth1</reference>
    <enable>true</enable>
    <metric-algorithm>etx</metric-algorithm>
    <split-horizon>>false</split-horizon>
  </interfaces>
  <interfaces>
    <reference>tun0</reference>
    <enable>true</enable>
    <metric-algorithm>two-out-of-three</metric-algorithm>
    <split-horizon>true</split-horizon>
  </interfaces>
</babel>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</config>

```

[A.4.](#) Configuring other Properties

<!-- In this example, two interfaces are configured for babeld

```

interface eth0
interface ppp0 hello-interval 30 update-interval 120

```

Here, ppp0 is a metered 3G link used for fallback connectivity. It runs with much higher than default time constants in order to avoid control traffic as much as possible.

-->

```

<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
    xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
      <enabled>true</enabled>
    </interface>
    <interface>
      <name>ppp0</name>
      <type>ianaift:ppp</type>
      <enabled>true</enabled>
    </interface>
  </interfaces>
  <routing
    xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type
          xmlns:babel=
            "urn:ietf:params:xml:ns:yang:ietf-babel">babel:babel
        </type>
        <name>name:babel</name>
        <babel
          xmlns="urn:ietf:params:xml:ns:yang:ietf-babel">
          <enable>true</enable>
          <interfaces>
            <reference>eth0</reference>
            <enable>true</enable>
            <metric-algorithm>two-out-of-three</metric-algorithm>
            <split-horizon>true</split-horizon>
          </interfaces>
          <interfaces>
            <reference>ppp0</reference>
            <enable>true</enable>
            <mcast-hello-interval>30</mcast-hello-interval>
            <update-interval>120</update-interval>
            <metric-algorithm>two-out-of-three</metric-algorithm>
          </interfaces>
        </babel>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>

```

Authors' Addresses

Mahesh Jethanandani
Kloud Services
California
USA

Email: mjethanandani@gmail.com

Barbara Stark
AT&T
Atlanta, GA
USA

Email: barbara.stark@att.com

