Network Working Group
Internet-Draft
Updates: 6126bis (if approved)
Intended status: Experimental
Expires: November 15, 2020

T. Bastian
Ecole Normale Superieure, Paris
J. Chroboczek
IRIF, University of Paris-Diderot

May 14, 2020

Announcing IPv4 routes with an IPv6 next-hop in the Babel routing protocol
draft-bastian-babel-v4ov6-01

Abstract

This document defines an extension to the Babel routing protocol that allows annoncing routes to an IPv4 prefix with an IPv6 next-hop, which makes it possible for IPv4 traffic to flow through interfaces that have not been assigned an IPv4 address.

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 November 15, 2020.

Table of Contents

1. Introduction

Traditionally, a routing table maps a network prefix of a given address family to a next-hop address in the same address family. The sole purpose of this next-hop address is to serve as an input to a protocol that will map it to a link-layer address, Neighbour Discovery (ND) [RFC4861] in the case of IPv6, Address Resolution (ARP) [RFC0826] in the case of IPv4. Therefore, there is no reason why the address family of the next hop address should match that of the prefix being announced: an IPv6 next-hop yields a link-layer address that is suitable for forwarding both IPv6 or IPv4 traffic.

We call a route towards an IPv4 prefix that uses an IPv6 next hop a "v4-over-v6" route. Since an IPv6 next-hop can use a link-local address that is autonomously configured, the use of v4-over-v6 routes enables a mode of operation where the network core has no statically assigned IP addresses of either family, thus significantly reducing the amount of manual configuration.

This document describes an extension that allows the Babel routing protocol [RFC6126bis] to announce routes towards IPv6 prefixes with IPv4 next hops. The extension is inspired by a previously defined extension to the BGP protocol [RFC5549].

1.1. Specification of Requirements

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.

2. Protocol operation

The Babel protocol fully supports double-stack operation: all data that represent a neighbour address or a network prefix are tagged by an Address Encoding (AE), a small integer that identifies the address family (IPv4 or IPv6) of the address of prefix, and describes how it is encoded. This extension defines a new AE, called v4-over-v6, which has the same format as the existing AE for IPv4 addresses. This new AE is only allowed in TLVs that carry network prefixes: TLVs that carry a neighbour address use the normal encodings for IPv6 addresses.

2.1. Announcing v4-over-v6 routes

A Babel node that needs to announce an IPv4 route over an interface that has no assigned IPv4 address MAY make a v4-over-v6 announcement. In order to do so, it first establishes an IPv6 next-hop address in the usual manner (either by sending the Babel packet over IPv6, or by including a Next Hop TLV containing an IPv6 address); it then sends an Update with AE equal to TBD containing the IPv4 prefix being announced.

If the outgoing interface has been assigned an IPv4 address, then, in the interest of maximising compatibility with existing routers, the sender SHOULD prefer an ordinary IPv4 announcement; even in that case, however, it MAY use a v4-over-v6 announcement. A node SHOULD NOT send both ordinary IPv4 and v4-over-v6 annoucements for the same prefix over a single interface (if the update is sent to a multicast address) or to a single neighbour (if sent to a unicast address), since doing that doubles the amount of routing traffic while providing no benefit.

2.2. Receiving v4-over-v6 routes

Upon reception of an Update TLV with a v4-over-v6 AE, a Babel node computes the IPv6 next-hop, as described in Section 4.6.9 of [RFC6126bis]. If no IPv6 next-hop exists, then the Update MUST be silently ignored. If an IPv6 next-hop exists, then the node MAY acquire the route being announced, as described in Section 3.5.3 of [RFC6126bis]; the parameters of the route are as follows:

- the prefix, plen, router-id, seqno, metric MUST be computed as for an IPv4 route, as described in Section 4.6.9 of [RFC6126bis];

- the next-hop MUST be computed as for an IPv6 route, as described in Section 4.6.9 of [RFC6126bis]: it is taken from the last preceding Next-Hop TLV with an AE field equal to 2 or 3; if no such entry exists, and if the Update TLV has been sent in a Babel packet carried over IPv6, then the next-hop is the network-layer source address of the packet.

As usual, a node MAY ignore the update, e.g., due to filtering (Appendix C of [RFC6126bis]). If a node cannot install v4-over-v6 routes, eg., due to hardware or software limitations, then routes to an IPv4 prefix with an IPv6 next-hop MUST NOT be selected, as described in Section 3.5.3 of [RFC6126bis].

2.3. Prefix and seqno requests

Prefix and seqno requests are used to request an update for a given prefix. Since they are not related to a specific Next-Hop, there is no semantic difference between ordinary IPv4 and v4-over-v6 requests.

A node SHOULD NOT send requests of either kind with the AE field being set to TBD (v4-over-v6); instead, it SHOULD request IPv4 updates using requests with the AE field being set to 1 (IPv4).

When receiving requests, AEs 1 (IPv4) and TBD (v4-over-v6) MUST be treated in the same manner: the receiver processes the request as described in Section 3.8 of [RFC6126bis]. If an Update is sent, then it MAY be sent with AE 1 or TBD, as described in Section 2.1 above, irrespective of which AE was used in the request.

When receiving a request with AE 0 (wildcard), the receiver SHOULD send a full route dump, as described in Section 3.8.1.1 of [RFC6126bis]. Any IPv4 routes contained in the route dump MAY use either AE 1 or AE TBD, as described in Section 2.1 above.

2.4. Other TLVs

The only other TLV defined by [RFC6126bis] that carries an AE field is the IHU TLV. IHU TLVs MUST NOT carry the AE TBD (v4-over-v6).

3. Backwards compatibility

This protocol extension adds no new TLVs or sub-TLVs.

This protocol extension uses a new AE. As discussed in Appendix D of [RFC6126bis] and specified in the same document, implementations that

do not understand the present extension will silently ignore the various TLVs that use this new AE. As a result, incompatible versions will ignore v4-over-v6 routes. They will also ignore requests with AE TBD, which, as stated in Section 2.3, are NOT RECOMMENDED.

Using a new AE introduces a new compression state, used to parse the network prefixes. As this compression state is separate from other AEs' states, it will not interfere with the compression state of unextended nodes.

This extension reuses the next-hop state from AEs 2 and 3 (IPv6), but makes no changes to the way it is updated, and therefore causes no compatibility issues.

As mentioned in Section 2.1, ordinary IPv4 announcements are preferred to v4-over-v6 announcements when the outgoing interface has an assigned IPv4 address; doing otherwise would prevent routers that do not implement this extension from learning the route being announced.

4. Protocol encoding

This extension defines the v4-over-v6 AE, whose value is TBD. This AE is solely used to tag network prefixes, and MUST NOT be used to tag peers' addresses, eg. in Next-Hop or IHU TLVs.

This extension defines no new TLVs or sub-TLVs.

4.1. Prefix encoding

Network prefixes tagged with AE TBD MUST be encoded and decoded as prefixes tagged with AE 1 (IPv4), as described in Section 4.3.1 of [RFC6126bis].

A new compression state for AE TBD (v4-over-v6) distinct from that of AE 1 (IPv4) is introduced, and MUST be used for address compression of prefixes tagged with AE TBD, as described in Section 4.6.9 of [RFC6126bis]

4.2. Changes for existing TLVs

The following TLVs MAY be tagged with AE TBD:

- Update (Type = 8)
- Route Request (Type = 9)

- Seqno Request (Type = 10)

As AE TBD is suitable only to tag network prefixes, IHU (Type = 5) and Next-Hop (Type = 7) TLVs MUST NOT be tagged with AE TBD. Such TLVs MUST be silently ignored.

4.2.1. Update

An Update (Type = 8) TLV with AE = TBD is constructed as described in Section 4.6.9 of [RFC6126bis] for AE 1 (IPv4), with the following specificities:

- Prefix. The Prefix field is constructed according to the Section 4.1 above.

- Next hop. The next hop is determined as described in Section 2.2 above.

4.2.2. Other valid TLVs tagged with AE = TBD

Any other valid TLV tagged with AE = TBD MUST be constructed and decoded as described in Section 4.6 of [RFC6126bis]. Network prefixes within MUST be constructed and decoded as described in Section 4.1 above.

5. IANA Considerations

IANA is requested to allocate a value (4 suggested) in the "Babel Address Encodings" registry as follows:

AE	Name	Reference
TBD	v4-over-v6	(this document)

6. Security Considerations

This extension does not fundamentally change the security properties of the Babel protocol: as described in Section 6 of [RFC6126bis], Babel must be protected by a suitable cryptographic mechanism in order to be made secure.

However, enabling this extension will allow IPv4 traffic to flow through sections of a network that have not been assigned IPv4 addresses, which, in turn, might allow IPv4 traffic to reach areas of the network that were previously inaccessible to such traffic. If this is undesirable, the flow of IPv4 traffic must be restricted by

the use of suitable filtering rules (Appendix C of [RFC6126bis]) together with matching access control rules in the data plane.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.

[RFC6126bis] Chroboczek, J. and D. Schinazi, "The Babel Routing Protocol", draft-ietf-babel-rfc6126bis-17 (work in progress), February 2020.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017.

7.2. Informative References

[RFC0826] Plummer, D., "An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, DOI 10.17487/RFC0826, November 1982.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007.

[RFC5549] Le Faucheur, F. and E. Rosen, "Advertising IPv4 Network Layer Reachability Information with an IPv6 Next Hop", RFC 5549, DOI 10.17487/RFC5549, May 2009.

Authors' Addresses

Theophile Bastian
Ecole Normale Superieure, Paris
France

Email: theophile.bastian@ens.fr

Juliusz Chroboczek
IRIF, University of Paris-Diderot
Case 7014
75205 Paris Cedex 13
France

Email: jch@irif.fr

Babel routing protocol
Internet-Draft
Intended status: Informational
Expires: April 11, 2020

B. Stark
AT&T
M. Jethanandani
VMware

October 9, 2019

Babel Information Model
draft-ietf-babel-information-model-10

Abstract

This Babel Information Model provides structured data elements for a Babel implementation reporting its current state and may allow limited configuration of some such data elements. This information model can be used as a basis for creating data models under various data modeling regimes.

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 April 11, 2020.

Table of Contents

1. Introduction

Babel is a loop-avoiding distance-vector routing protocol defined in [I-D.ietf-babel-rfc6126bis]. [I-D.ietf-babel-hmac] defines a security mechanism that allows Babel packets to be cryptographically authenticated, and [I-D.ietf-babel-dtls] defines a security mechanism that allows Babel packets to be encrypted. This document describes an information model for Babel (including implementations using one or both of these security mechanisms) that can be used to create management protocol data models (such as a NETCONF [RFC6241] YANG [RFC7950] data model.

Due to the simplicity of the Babel protocol, most of the information model is focused on reporting Babel protocol operational state, and very little of that is considered mandatory to implement for an implementation claiming compliance with this information model. Some parameters may be configurable. However, it is up to the Babel implementation whether to allow any of these to be configured within its implementation. Where the implementation does not allow

configuration of these parameters, it MAY still choose to expose them as read-only.

The Information Model is presented using a hierarchical structure. This does not preclude a data model based on this Information Model from using a referential or other structure.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] and updated by [RFC8174].

1.2. Notation

This document uses a programming language-like notation to define the properties of the objects of the information model. An optional property is enclosed by square brackets, [ ], and a list property is indicated by two numbers in angle brackets, <m..n>, where m indicates the minimal number of list elements, and n indicates the maximum number of list elements. The symbol * for n means there are no defined limits on the number of list elements. Each parameter and object includes an indication of "ro" or "rw". "ro" means the parameter or object is read-only. "rw" means it is read-write. For an object, read-write means instances of the object can be created or deleted. If an implementation is allowed to choose to implement a "rw" parameter as read-only, this is noted in the parameter description.

The object definitions use base types that are defined as follows:

binary — A binary string (sequence of octets).

boolean — A type representing a Boolean (true or false) value.

counter — A non-negative integer that monotonically increases. Counters may have discontinuities and they are not expected to persist across restarts.

datetime — A type representing a date and time using the Gregorian calendar. The datetime format MUST conform to RFC 3339 [RFC3339].

ip-address — A type representing an IP address. This type supports both IPv4 and IPv6 addresses.

operation A type representing a remote procedure call or other action that can be used to manipulate data elements or system behaviors.

reference A type representing a reference to another information or data model element or to some other device resource.

string A type representing a human-readable string consisting of a (possibly restricted) subset of Unicode and ISO/IEC 10646 [ISO.10646] characters.

uint A type representing an unsigned integer number. This information model does not define a precision.

2. Overview

The Information Model is hierarchically structured as follows:

```
+-- babel-information
   +-- babel-implementation-version
   +-- babel-enable
   +-- router-id
   +-- self-seqno
   +-- babel-metric-comp-algorithms
   +-- babel-security-supported
   +-- babel-mac-algorithms
   +-- babel-dtls-cert-types
   +-- babel-stats-enable
   +-- babel-stats-reset
   +-- babel-constants
   |  +-- babel-udp-port
   |  +-- babel-mcast-group
   +-- babel-interfaces
   |  +-- babel-interface-reference
   |  +-- babel-interface-enable
   |  +-- babel-interface-metric-algorithm
   |  +-- babel-interface-split-horizon
   |  +-- babel-mcast-hello-seqno
   |  +-- babel-mcast-hello-interval
   |  +-- babel-update-interval
   |  +-- babel-mac-enable
   |  +-- babel-if-mac-key-sets
   |  +-- babel-mac-verify
   |  +-- babel-dtls-enable
   |  +-- babel-if-dtls-cert-sets
   |  +-- babel-dtls-cached-info
   |  +-- babel-dtls-cert-prefer
   |  +-- babel-packet-log-enable
```

```
      |  +-- babel-packet-log
      |  +-- babel-if-stats
      |  |  +-- babel-sent-mcast-hello
      |  |  +-- babel-sent-mcast-update
      |  |  +-- babel-sent-ucast-hello
      |  |  +-- babel-sent-ucast-update
      |  |  +-- babel-sent-IHU
      |  |  +-- babel-received-packets
      |  +-- babel-neighbors
      |  |  +-- babel-neighbor-address
      |  |  +-- babel-hello-mcast-history
      |  |  +-- babel-hello-ucast-history
      |  |  +-- babel-txcost
      |  |  +-- babel-exp-mcast-hello-seqno
      |  |  +-- babel-exp-ucast-hello-seqno
      |  |  +-- babel-ucast-hello-seqno
      |  |  +-- babel-ucast-hello-interval
      |  |  +-- babel-rxcost
      |  |  +-- babel-cost
      +-- babel-routes
      |  +-- babel-route-prefix
      |  +-- babel-route-prefix-length
      |  +-- babel-route-router-id
      |  +-- babel-route-neighbor
      |  +-- babel-route-received-metric
      |  +-- babel-route-calculated-metric
      |  +-- babel-route-seqno
      |  +-- babel-route-next-hop
      |  +-- babel-route-feasible
      |  +-- babel-route-selected
      +-- babel-mac-key-sets
      |  +-- babel-mac-default-apply
      |  +-- babel-mac-keys
      |  |  +-- babel-mac-key-name
      |  |  +-- babel-mac-key-use-sign
      |  |  +-- babel-mac-key-use-verify
      |  |  +-- babel-mac-key-value
      |  |  +-- babel-mac-key-algorithm
      |  |  +-- babel-mac-key-test
      +-- babel-dtls-cert-sets
      |  +-- babel-dtls-default-apply
      |  +-- babel-dtls-certs
      |  |  +-- babel-cert-name
      |  |  +-- babel-cert-value
      |  |  +-- babel-cert-type
      |  |  +-- babel-cert-private-key
      |  |  +-- babel-cert-test
```

Most parameters are read-only. Following is a descriptive list of the parameters that are not required to be read-only:

- enable/disable Babel
- create/delete Babel MAC Key sets
- create/delete Babel DTLS Certificate sets
- enable/disable statistics collection
- Constant: UDP port
- Constant: IPv6 multicast group
- Interface: Metric algorithm
- Interface: Split horizon
- Interface: enable/disable Babel on this interface
- Interface: sets of MAC keys
- Interface: MAC algorithm
- Interface: verify received MAC packets
- Interface: set of DTLS certificates
- Interface: use cached info extensions
- Interface: preferred order of certificate types
- Interface: enable/disable packet log
- MAC-keys: create/delete entries
- MAC-keys: key used to sign packets
- MAC-keys: key used to verify packets
- DTLS-certs: create/delete entries

The following parameters are required to return no value when read:

- MAC key values
- DTLS certificate values

Note that this overview is intended simply to be informative and is not normative. If there is any discrepancy between this overview and the detailed information model definitions in subsequent sections, the error is in this overview.

3. The Information Model

3.1. Definition of babel-information-obj

```
 object {
      string                   ro babel-implementation-version;
      boolean                  rw babel-enable;
      binary                   ro babel-self-router-id;
     [uint                     ro babel-self-seqno;]
      string                   ro babel-metric-comp-algorithms<1..*>;
      string                   ro babel-security-supported<0..*>;
     [string                   ro babel-mac-algorithms<1..*>;]
     [string                   ro babel-dtls-cert-types<1..*>;]
     [boolean                  rw babel-stats-enable;]
     [operation                   babel-stats-reset;]
      babel-constants-obj      ro babel-constants;
      babel-interfaces-obj     ro babel-interfaces<0..*>;
      babel-routes-obj         ro babel-routes<0..*>;
     [babel-mac-key-sets-obj   rw babel-mac-key-sets<0..*>;]
     [babel-dtls-cert-sets-obj rw babel-dtls-cert-sets<0..*>;]
  } babel-information-obj;
```

babel-implementation-version: The name and version of this implementation of the Babel protocol.

babel-enable: When written, it configures whether the protocol should be enabled (true) or disabled (false). A read from the running or intended datastore indicates the configured administrative value of whether the protocol is enabled (true) or not (false). A read from the operational datastore indicates whether the protocol is actually running (true) or not (i.e., it indicates the operational state of the protocol). A data model that does not replicate parameters for running and operational datastores can implement this as two separate parameters. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-self-router-id: The router-id used by this instance of the Babel protocol to identify itself. [I-D.ietf-babel-rfc6126bis] describes this as an arbitrary string of 8 octets. The router-id value MUST NOT consist of all zeroes or all ones.

babel-self-seqno: The current sequence number included in route updates for routes originated by this node. This is a 16-bit unsigned integer.

babel-metric-comp-algorithms: List of supported cost computation algorithms. Possible values include "2-out-of-3", and "ETX". "2-out-of-3" is described in [I-D.ietf-babel-rfc6126bis], section A.2.1. "ETX" is described in [I-D.ietf-babel-rfc6126bis], section A.2.2.

babel-security-supported: List of supported security mechanisms. Possible values include "MAC" and "DTLS".

babel-mac-algorithms: List of supported MAC computation algorithms. Possible values include "HMAC-SHA256", "BLAKE2s".

babel-dtls-cert-types: List of supported DTLS certificate types. Possible values include "X.509" and "RawPublicKey".

babel-stats-enable: Indicates whether statistics collection is enabled (true) or disabled (false) on all interfaces.

babel-stats-reset: An operation that resets all babel-if-stats parameters to zero. This operation has no input or output parameters.

babel-constants: A babel-constants-obj object.

babel-interfaces: A set of babel-interface-obj objects.

babel-routes: A set of babel-route-obj objects. Contains the routes known to this node.

babel-mac-key-sets: A babel-mac-key-sets-obj object. If this object is implemented, it provides access to parameters related to the MAC security mechanism. An implementation MAY choose to expose this object as read-only ("ro").

babel-dtls-cert-sets: A babel-dtls-cert-sets-obj object. If this object is implemented, it provides access to parameters related to the DTLS security mechanism. An implementation MAY choose to expose this object as read-only ("ro").

3.2. Definition of babel-constants-obj

```
     object {
          uint          rw babel-udp-port;
         [ip-address    rw babel-mcast-group;]
      } babel-constants-obj;
```

babel-udp-port: UDP port for sending and listening for Babel packets. Default is 6696. An implementation MAY choose to expose this parameter as read-only ("ro"). This is a 16-bit unsigned integer.

babel-mcast-group: Multicast group for sending and listening to multicast announcements on IPv6. Default is ff02::1:6. An implementation MAY choose to expose this parameter as read-only ("ro").

3.3. Definition of babel-interfaces-obj

```
     object {
          reference            ro babel-interface-reference;
         [boolean              rw babel-interface-enable;]
          string               rw babel-interface-metric-algorithm;
         [boolean              rw babel-interface-split-horizon;]
         [uint                 ro babel-mcast-hello-seqno;]
         [uint                 ro babel-mcast-hello-interval;]
         [uint                 ro babel-update-interval;]
         [boolean              rw babel-mac-enable;]
         [reference            rw babel-if-mac-key-sets<0..*>;]
         [boolean              rw babel-mac-verify;]
         [boolean              rw babel-dtls-enable;]
         [reference            rw babel-if-dtls-cert-sets<0..*>;]
         [boolean              rw babel-dtls-cached-info;]
         [string               rw babel-dtls-cert-prefer<0..*>;]
         [boolean              rw babel-packet-log-enable;]
         [reference            ro babel-packet-log;]
         [babel-if-stats-obj   ro babel-if-stats;]
          babel-neighbors-obj  ro babel-neighbors<0..*>;
      } babel-interfaces-obj;
```

babel-interface-reference: Reference to an interface object that can be used to send and receive IPv6 packets, as defined by the data model (e.g., YANG [RFC7950], BBF [TR-181]). Referencing syntax will be specific to the data model. If there is no set of interface objects available, this should be a string that indicates the interface name used by the underlying operating system.

babel-interface-enable: When written, it configures whether the protocol should be enabled (true) or disabled (false) on this

interface. A read from the running or intended datastore indicates the configured administrative value of whether the protocol is enabled (true) or not (false). A read from the operational datastore indicates whether the protocol is actually running (true) or not (i.e., it indicates the operational state of the protocol). A data model that does not replicate parameters for running and operational datastores can implement this as two separate parameters. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-interface-metric-algorithm: Indicates the metric computation algorithm used on this interface. The value MUST be one of those listed in the babel-information-obj babel-metric-comp-algorithms parameter. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-interface-split-horizon: 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. Split horizon optimization is described in [I-D.ietf-babel-rfc6126bis], section 3.7.4. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-mcast-hello-seqno: The current sequence number in use for multicast Hellos sent on this interface. This is a 16-bit unsigned integer.

babel-mcast-hello-interval: The current interval in use for multicast Hellos sent on this interface. Units are centiseconds. This is a 16-bit unsigned integer.

babel-update-interval: The current interval in use for all updates (multicast and unicast) sent on this interface. Units are centiseconds. This is a 16-bit unsigned integer.

babel-mac-enable: Indicates whether the MAC security mechanism is enabled (true) or disabled (false). An implementation MAY choose to expose this parameter as read-only ("ro").

babel-if-mac-keys-sets: List of references to the babel-mac entries that apply to this interface. When an interface instance is created, all babel-mac-key-sets instances with babel-mac-default-apply "true" will be included in this list. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-mac-verify 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. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-dtls-enable: Indicates whether the DTLS security mechanism is enabled (true) or disabled (false). An implementation MAY choose to expose this parameter as read-only ("ro").

babel-if-dtls-cert-sets: List of references to the babel-dtls-cert-sets entries that apply to this interface. When an interface instance is created, all babel-dtls-cert-sets instances with babel-dtls-default-apply "true" will be included in this list. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-dtls-cached-info: Indicates whether the cached_info extension is included in ClientHello and ServerHello packets. The extension is included if the value is "true". An implementation MAY choose to expose this parameter as read-only ("ro").

babel-dtls-cert-prefer: List of supported certificate types, in order of preference. The values MUST be among those listed in the babel-dtls-cert-types parameter. 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 babel-dtls-certs object of a referenced babel-dtls instance and that have a non-empty babel-cert-private-key will be used to populate the client_certificate_type extension in a Client Hello.

babel-packet-log-enable: Indicates whether packet logging is enabled (true) or disabled (false) on this interface.

babel-packet-log: A reference or url link to a file that contains a timestamped log of packets received and sent on babel-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 babel-packet-log-enable.

babel-if-stats: Statistics collection object for this interface.

babel-neighbors: A set of babel-neighbors-obj objects.

3.4. Definition of babel-if-stats-obj

```
     object {
          uint                ro babel-sent-mcast-hello;
          uint                ro babel-sent-mcast-update;
          uint                ro babel-sent-ucast-hello;
          uint                ro babel-sent-ucast-update;
          uint                ro babel-sent-IHU;
          uint                ro babel-received-packets;
      } babel-if-stats-obj;
```

babel-sent-mcast-hello: A count of the number of multicast Hello packets sent on this interface.

babel-sent-mcast-update: A count of the number of multicast update packets sent on this interface.

babel-sent-ucast-hello: A count of the number of unicast Hello packets sent on this interface.

babel-sent-ucast-update: A count of the number of unicast update packets sent on this interface.

babel-sent-IHU: A count of the number of IHU packets sent on this interface.

babel-received-packets: A count of the number of Babel packets received on this interface.

3.5. Definition of babel-neighbors-obj

```
     object {
          ip-address          ro babel-neighbor-address;
         [binary              ro babel-hello-mcast-history;]
         [binary              ro babel-hello-ucast-history;]
          uint                ro babel-txcost;
          uint                ro babel-exp-mcast-hello-seqno;
          uint                ro babel-exp-ucast-hello-seqno;
         [uint                ro babel-ucast-hello-seqno;]
         [uint                ro babel-ucast-hello-interval;]
         [uint                ro babel-rxcost;]
         [uint                ro babel-cost;]
      } babel-neighbors-obj;
```

babel-neighbor-address: IPv4 or IPv6 address the neighbor sends packets from.

babel-hello-mcast-history: The multicast Hello history of whether or not the multicast Hello packets prior to babel-exp-mcast-hello-seqno were received. A binary sequence where the most recently

received Hello is expressed as a "1" placed in the left-most bit, with prior bits shifted right (and "0" bits placed between prior Hello bits and most recent Hello for any not-received Hellos). This value should be displayed using hex digits ([0-9a-fA-F]). See [I-D.ietf-babel-rfc6126bis], section A.1.

babel-hello-ucast-history: The unicast Hello history of whether or not the unicast Hello packets prior to babel-exp-ucast-hello-seqno were received. A binary sequence where the most recently received Hello is expressed as a "1" placed in the left-most bit, with prior bits shifted right (and "0" bits placed between prior Hello bits and most recent Hello for any not-received Hellos). This value should be displayed using hex digits ([0-9a-fA-F]). See [I-D.ietf-babel-rfc6126bis], section A.1.

babel-txcost: Transmission cost value from the last IHU packet received from this neighbor, or maximum value to indicate the IHU hold timer for this neighbor has expired. See [I-D.ietf-babel-rfc6126bis], section 3.4.2. This is a 16-bit unsigned integer.

babel-exp-mcast-hello-seqno: 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. This is a 16-bit unsigned integer; if the data model uses zero (0) to represent NULL values for unsigned integers, the data model MAY use a different data type that allows differentiation between zero (0) and NULL.

babel-exp-ucast-hello-seqno: 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. This is a 16-bit unsigned integer; if the data model uses zero (0) to represent NULL values for unsigned integers, the data model MAY use a different data type that allows differentiation between zero (0) and NULL.

babel-ucast-hello-seqno: The current sequence number in use for unicast Hellos sent to this neighbor. If unicast Hellos are not being sent, this MUST be NULL. This is a 16-bit unsigned integer; if the data model uses zero (0) to represent NULL values for unsigned integers, the data model MAY use a different data type that allows differentiation between zero (0) and NULL.

babel-ucast-hello-interval: The current interval in use for unicast Hellos sent to this neighbor. Units are centiseconds. This is a 16-bit unsigned integer.

babel-rxcost: 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. See [I-D.ietf-babel-rfc6126bis], section 3.4.3. This is a 16-bit unsigned integer.

babel-cost: The link cost, as 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. This is a 16-bit unsigned integer.

3.6. Definition of babel-routes-obj

```
  object {
       ip-address           ro babel-route-prefix;
       uint                 ro babel-route-prefix-length;
       binary               ro babel-route-router-id;
       string               ro babel-route-neighbor;
       uint                 ro babel-route-received-metric;
       uint                 ro babel-route-calculated-metric;
       uint                 ro babel-route-seqno;
       ip-address           ro babel-route-next-hop;
       boolean              ro babel-route-feasible;
       boolean              ro babel-route-selected;
   } babel-routes-obj;
```

babel-route-prefix: Prefix (expressed in IP address format) for which this route is advertised.

babel-route-prefix-length: Length of the prefix for which this route is advertised.

babel-route-router-id: The router-id of the router that originated this route.

babel-route-neighbor: Reference to the babel-neighbors entry for the neighbor that advertised this route.

babel-route-received-metric: The metric with which this route was advertised by the neighbor, or maximum value to indicate the route was recently retracted and is temporarily unreachable (see Section 3.5.5 of [I-D.ietf-babel-rfc6126bis]). This metric will be NULL if the route was not received from a neighbor but was generated through other means. At least one of babel-route-calculated-metric and babel-route-received-metric MUST be non-NULL. Having both be non-NULL is expected for a route that is received and subsequently advertised. This is a 16-bit unsigned

integer; if the data model uses zero (0) to represent NULL values for unsigned integers, the data model MAY use a different data type that allows differentiation between zero (0) and NULL.

babel-route-calculated-metric: A calculated metric for this route. How the metric is calculated is implementation-specific. Maximum value indicates the route was recently retracted and is temporarily unreachable (see Section 3.5.5 of [I-D.ietf-babel-rfc6126bis]). At least one of babel-route-calculated-metric and babel-route-received-metric MUST be non-NULL. Having both be non-NULL is expected for a route that is received and subsequently advertised. This is a 16-bit unsigned integer; if the data model uses zero (0) to represent NULL values for unsigned integers, the data model MAY use a different data type that allows differentiation between zero (0) and NULL.

babel-route-seqno: The sequence number with which this route was advertised. This is a 16-bit unsigned integer.

babel-route-next-hop: The next-hop address of this route. This will be empty if this route has no next-hop address.

babel-route-feasible: A Boolean flag indicating whether this route is feasible, as defined in Section 3.5.1 of [I-D.ietf-babel-rfc6126bis]).

babel-route-selected: A Boolean flag indicating whether this route is selected (i.e., whether it is currently being used for forwarding and is being advertised).

3.7. Definition of babel-mac-key-sets-obj

```
  object {
       boolean               rw babel-mac-default-apply;
       babel-mac-keys-obj    rw babel-mac-keys<0..*>;
   } babel-mac-obj;
```

babel-mac-default-apply: A Boolean flag indicating whether this babel-mac instance is applied to all new babel-interface instances, by default. If "true", this instance is applied to new babel-interfaces instances at the time they are created, by including it in the babel-interface-mac-keys list. If "false", this instance is not applied to new babel-interfaces instances when they are created. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-mac-keys: A set of babel-mac-keys-obj objects.

3.8. Definition of babel-mac-keys-obj

```
     object {
          string                rw babel-mac-key-name;
          boolean               rw babel-mac-key-use-sign;
          boolean               rw babel-mac-key-use-verify;
          binary                -- babel-mac-key-value;
          string                rw babel-mac-key-algorithm;
         [operation                babel-mac-key-test;]
      } babel-mac-keys-obj;
```

babel-mac-key-name: 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 MUST NOT be empty and can only be provided when this instance is created (i.e., it is not subsequently writable). The value MAY be auto-generated if not explicitly supplied when the instance is created.

babel-key-use-sign: 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. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-key-use-verify: 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 with the MAC in an incoming packet. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-key-value: 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 when read, 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 babel-mac-key-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].

babel-mac-key-algorithm The name of the MAC algorithm used with this key. The value MUST be the same as one of the enumerations listed in the babel-mac-algorithms parameter. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-mac-test: 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 babel-mac-key-value and the babel-mac-algorithm. The output of this operation is the resulting hash, as a binary string.

3.9. Definition of babel-dtls-cert-sets-obj

```
object {
     boolean               rw babel-dtls-default-apply;
     babel-dtls-certs-obj  rw babel-dtls-certs<0..*>;
} babel-dtls-obj;
```

babel-dtls-default-apply: A Boolean flag indicating whether this babel-dtls instance is applied to all new babel-interface instances, by default. If "true", this instance is applied to new babel-interfaces instances at the time they are created, by including it in the babel-interface-dtls-certs list. If "false", this instance is not applied to new babel-interfaces instances when they are created. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-dtls-certs: A set of babel-dtls-keys-obj objects. This contains both certificates for this implementation to present for authentication, and to accept from others. Certificates with a non-empty babel-cert-private-key can be presented by this implementation for authentication.

3.10. Definition of babel-dtls-certs-obj

```
object {
     string                rw babel-cert-name;
     string                rw babel-cert-value;
     string                rw babel-cert-type;
     binary                -- babel-cert-private-key;
    [operation                babel-cert-test;]
} babel-dtls-certs-obj;
```

babel-cert-name: 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). The value MAY be auto-generated if not explicitly supplied when the instance is created.

babel-cert-value: The DTLS certificate in PEM format [RFC7468]. This value MUST be provided when this instance is created, and is not subsequently writable.

babel-cert-type: The name of the certificate type of this object instance. The value MUST be the same as one of the enumerations listed in the babel-dtls-cert-types parameter. This value can only be provided when this instance is created, and is not subsequently writable.

babel-cert-private-key: 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 when read, or through permissions, or other means. This value can only be provided when this instance is created, and is not subsequently writable.

babel-cert-test: An operation that allows a hash of the provided input string to be created using the certificate public key and the SHA-256 hash algorithm. Input to this operation is a binary string. The output of this operation is the resulting hash, as a binary string.

4. Extending the Information Model

Implementations MAY extend this information model with other parameters or objects. For example, an implementation MAY choose to expose Babel route filtering rules by adding a route filtering object with parameters appropriate to how route filtering is done in that implementation. The precise means used to extend the information model would be specific to the data model the implementation uses to expose this information.

5. Security Considerations

This document defines a set of information model objects and parameters that may be exposed to be visible from other devices, and some of which may be configured. Securing access to and ensuring the integrity of this data is in scope of and the responsibility of any data model derived from this information model. Specifically, any YANG [RFC7950] data model is expected to define security exposure of the various parameters, and a [TR-181] data model will be secured by the mechanisms defined for the management protocol used to transport it.

Misconfiguration (whether unintentional or malicious) can prevent reachability or cause poor network performance (increased latency,

jitter, etc.). The information in this model discloses network topology, which can be used to mount subsequent attacks on traffic traversing the network.

This information model defines objects that can allow credentials (for this device, for trusted devices, and for trusted certificate authorities) to be added and deleted. Public keys may be exposed through this model. This model requires that private keys never be exposed. The Babel security mechanisms that make use of these credentials (e.g., [I-D.ietf-babel-dtls], [I-D.ietf-babel-hmac]) identify what credentials can be used with those mechanisms.

MAC keys are allowed to be as short as zero-length. This is useful for testing. Network operators are advised to follow current best practices for key length and generation of keys related to the MAC algorithm associated with the key. Short (and zero-length) keys and keys that make use of only alphanumeric characters are highly susceptible to brute force attacks.

6. IANA Considerations

This document has no IANA actions.

7. Acknowledgements

Juliusz Chroboczek, Toke Hoeiland-Joergensen, David Schinazi, Acee Lindem, and Carsten Bormann have been very helpful in refining this information model.

The language in the Notation section was mostly taken from [RFC8193].

8. References

8.1. Normative References

[I-D.ietf-babel-rfc6126bis]
Chroboczek, J. and D. Schinazi, "The Babel Routing Protocol", draft-ietf-babel-rfc6126bis-14 (work in progress), August 2019.

[libpcap] Wireshark, "Libpcap File Format", 2015, <https://wiki.wireshark.org/Development/LibpcapFileFormat>.

[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>.

[RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <https://www.rfc-editor.org/info/rfc7468>.

[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>.

8.2. Informative References

[I-D.ietf-babel-dtls]
Decimo, A., Schinazi, D., and J. Chroboczek, "Babel Routing Protocol over Datagram Transport Layer Security", draft-ietf-babel-dtls-09 (work in progress), August 2019.

[I-D.ietf-babel-hmac]
Do, C., Kolodziejak, W., and J. Chroboczek, "MAC authentication for the Babel routing protocol", draft-ietf-babel-hmac-10 (work in progress), August 2019.

[ISO.10646]
International Organization for Standardization, "Information Technology - Universal Multiple-Octet Coded Character Set (UCS)", ISO Standard 10646:2014, 2014.

[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>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[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>.

[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>.

[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>.

[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>.

[RFC8193] Burbridge, T., Eardley, P., Bagnulo, M., and J. Schoenwaelder, "Information Model for Large-Scale Measurement Platforms (LMAPs)", RFC 8193, DOI 10.17487/RFC8193, August 2017, <https://www.rfc-editor.org/info/rfc8193>.

[TR-181] Broadband Forum, "Device Data Model", <http://cwmp-data-models.broadband-forum.org/>.

Authors' Addresses

Barbara Stark
AT&T
Atlanta, GA
US

Email: barbara.stark@att.com

Mahesh Jethanandani
VMware
California
US

Email: mjethanandani@gmail.com

Babel routing protocol
Internet-Draft
Intended status: Informational
Expires: 30 July 2021

B.H. Stark
AT&T
M.J. Jethanandani
VMware

26 January 2021

Babel Information Model
draft-ietf-babel-information-model-12

Abstract

This Babel Information Model provides structured data elements for a Babel implementation reporting its current state and may allow limited configuration of some such data elements. This information model can be used as a basis for creating data models under various data modeling regimes. This information model only includes parameters and parameter values useful for managing Babel over IPv6.

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 30 July 2021.

Table of Contents

1. Introduction

Babel is a loop-avoiding distance-vector routing protocol defined in [I-D.ietf-babel-rfc6126bis]. [I-D.ietf-babel-hmac] defines a security mechanism that allows Babel packets to be cryptographically authenticated, and [I-D.ietf-babel-dtls] defines a security mechanism that allows Babel packets to be both authenticated and encrypted. This document describes an information model for Babel (including implementations using one or both of these security mechanisms) that can be used to create management protocol data models (such as a NETCONF [RFC6241] YANG [RFC7950] data model).

Due to the simplicity of the Babel protocol, most of the information model is focused on reporting Babel protocol operational state, and very little of that is considered mandatory to implement for an implementation claiming compliance with this information model. Some parameters may be configurable. However, it is up to the Babel implementation whether to allow any of these to be configured within its implementation. Where the implementation does not allow configuration of these parameters, it MAY still choose to expose them as read-only.

The Information Model is presented using a hierarchical structure. This does not preclude a data model based on this Information Model from using a referential or other structure.

This information model only includes parameters and parameter values useful for managing Babel over IPv6. This model has no parameters or values specific to operating Babel over IPv4, even though [I-D.ietf-babel-rfc6126bis] does define a multicast group for sending and listening to multicast announcements on IPv4. There is less likelihood of breakage due to inconsistent configuration and increased implementation simplicity if Babel is operated always and only over IPv6. Running Babel over IPv6 requires IPv6 at the link layer and does not need advertised prefixes, router advertisements or DHCPv6 to be present in the network. Link-local IPv6 is widely supported among devices where Babel is expected to be used. Note that Babel over IPv6 can be used for configuration of both IPv4 and IPv6 routes.

1.1. 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 BCP014 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Notation

This document uses a programming language-like notation to define the properties of the objects of the information model. An optional property is enclosed by square brackets, [ ], and a list property is indicated by two numbers in angle brackets, <m..n>, where m indicates the minimal number of list elements, and n indicates the maximum number of list elements. The symbol * for n means there are no defined limits on the number of list elements. Each parameter and object includes an indication of "ro" or "rw". "ro" means the parameter or object is read-only. "rw" means it is read-write. For an object, read-write means instances of the object can be created or

deleted. If an implementation is allowed to choose to implement a "rw" parameter as read-only, this is noted in the parameter description.

The object definitions use base types that are defined as follows:

binary — A binary string (sequence of octets).

boolean — A type representing a Boolean (true or false) value.

datetime — A type representing a date and time using the Gregorian calendar. The datetime format MUST conform to RFC 3339 [RFC3339] Section 5.6.

ip-address — A type representing an IP address. This type supports both IPv4 and IPv6 addresses.

operation — A type representing a remote procedure call or other action that can be used to manipulate data elements or system behaviors.

reference — A type representing a reference to another information or data model element or to some other device resource.

string — A type representing a human-readable string consisting of a (possibly restricted) subset of Unicode and ISO/IEC 10646 [ISO.10646] characters.

uint — A type representing an unsigned integer number. This information model does not define a precision.

2. Overview

The Information Model is hierarchically structured as follows:

```
+-- babel-information
   +-- babel-implementation-version
   +-- babel-enable
   +-- router-id
   +-- self-seqno
   +-- babel-metric-comp-algorithms
   +-- babel-security-supported
   +-- babel-mac-algorithms
   +-- babel-dtls-cert-types
   +-- babel-stats-enable
   +-- babel-stats-reset
   +-- babel-constants
   |  +-- babel-udp-port
```

```
      |  +-- babel-mcast-group
      +-- babel-interfaces
      |  +-- babel-interface-reference
      |  +-- babel-interface-enable
      |  +-- babel-interface-metric-algorithm
      |  +-- babel-interface-split-horizon
      |  +-- babel-mcast-hello-seqno
      |  +-- babel-mcast-hello-interval
      |  +-- babel-update-interval
      |  +-- babel-mac-enable
      |  +-- babel-if-mac-key-sets
      |  +-- babel-mac-verify
      |  +-- babel-dtls-enable
      |  +-- babel-if-dtls-cert-sets
      |  +-- babel-dtls-cached-info
      |  +-- babel-dtls-cert-prefer
      |  +-- babel-packet-log-enable
      |  +-- babel-packet-log
      |  +-- babel-if-stats
      |  |  +-- babel-sent-mcast-hello
      |  |  +-- babel-sent-mcast-update
      |  |  +-- babel-sent-ucast-hello
      |  |  +-- babel-sent-ucast-update
      |  |  +-- babel-sent-IHU
      |  |  +-- babel-received-packets
      |  +-- babel-neighbors
      |     +-- babel-neighbor-address
      |     +-- babel-hello-mcast-history
      |     +-- babel-hello-ucast-history
      |     +-- babel-txcost
      |     +-- babel-exp-mcast-hello-seqno
      |     +-- babel-exp-ucast-hello-seqno
      |     +-- babel-ucast-hello-seqno
      |     +-- babel-ucast-hello-interval
      |     +-- babel-rxcost
      |     +-- babel-cost
      +-- babel-routes
      |  +-- babel-route-prefix
      |  +-- babel-route-prefix-length
      |  +-- babel-route-router-id
      |  +-- babel-route-neighbor
      |  +-- babel-route-received-metric
      |  +-- babel-route-calculated-metric
      |  +-- babel-route-seqno
      |  +-- babel-route-next-hop
      |  +-- babel-route-feasible
      |  +-- babel-route-selected
      +-- babel-mac-key-sets
```

```
      |  +-- babel-mac-default-apply
      |  +-- babel-mac-keys
      |     +-- babel-mac-key-name
      |     +-- babel-mac-key-use-send
      |     +-- babel-mac-key-use-verify
      |     +-- babel-mac-key-value
      |     +-- babel-mac-key-algorithm
      |     +-- babel-mac-key-test
      +-- babel-dtls-cert-sets
         +-- babel-dtls-default-apply
         +-- babel-dtls-certs
            +-- babel-cert-name
            +-- babel-cert-value
            +-- babel-cert-type
            +-- babel-cert-private-key
```

Most parameters are read-only. Following is a descriptive list of the parameters that are not required to be read-only:

* enable/disable Babel

* create/delete Babel MAC Key sets

* create/delete Babel Certificate sets

* enable/disable statistics collection

* Constant: UDP port

* Constant: IPv6 multicast group

* Interface: enable/disable Babel on this interface

* Interface: Metric algorithm

* Interface: Split horizon

* Interface: sets of MAC keys

* Interface: verify received MAC packets

* Interface: set of certificates for use with DTLS

* Interface: use cached info extensions

* Interface: preferred order of certificate types

* Interface: enable/disable packet log

* MAC-keys: create/delete entries

* MAC-keys: key used for sent packets

* MAC-keys: key used to verify packets

* DTLS-certs: create/delete entries

The following parameters are required to return no value when read:

* MAC key values

* DTLS private keys

Note that this overview is intended simply to be informative and is not normative. If there is any discrepancy between this overview and the detailed information model definitions in subsequent sections, the error is in this overview.

3. The Information Model

3.1. Definition of babel-information-obj

```
object {
     string                    ro babel-implementation-version;
     boolean                   rw babel-enable;
     binary                    ro babel-self-router-id;
    [uint                      ro babel-self-seqno;]
     string                    ro babel-metric-comp-algorithms<1..*>;
     string                    ro babel-security-supported<0..*>;
    [string                    ro babel-mac-algorithms<1..*>;]
    [string                    ro babel-dtls-cert-types<1..*>;]
    [boolean                   rw babel-stats-enable;]
    [operation                    babel-stats-reset;]
     babel-constants-obj       ro babel-constants;
     babel-interface-obj       ro babel-interfaces<0..*>;
     babel-route-obj           ro babel-routes<0..*>;
    [babel-mac-key-set-obj     rw babel-mac-key-sets<0..*>;]
    [babel-dtls-cert-set-obj   rw babel-dtls-cert-sets<0..*>;]
} babel-information-obj;
```

babel-implementation-version: The name and version of this implementation of the Babel protocol.

babel-enable: When written, it configures whether the protocol should be enabled (true) or disabled (false). A read from the running or intended datastore indicates the configured administrative value of whether the protocol is enabled (true) or

not (false). A read from the operational datastore indicates whether the protocol is actually running (true) or not (i.e., it indicates the operational state of the protocol). A data model that does not replicate parameters for running and operational datastores can implement this as two separate parameters. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-self-router-id: The router-id used by this instance of the Babel protocol to identify itself. [I-D.ietf-babel-rfc6126bis] describes this as an arbitrary string of 8 octets. The router-id value MUST NOT consist of all zeroes or all ones.

babel-self-seqno: The current sequence number included in route updates for routes originated by this node. This is a 16-bit unsigned integer.

babel-metric-comp-algorithms: List of supported cost computation algorithms. Possible values include "2-out-of-3", and "ETX". "2-out-of-3" is described in [I-D.ietf-babel-rfc6126bis], section A.2.1. "ETX" is described in [I-D.ietf-babel-rfc6126bis], section A.2.2.

babel-security-supported: List of supported security mechanisms. Possible values include "MAC" to indicate support of [I-D.ietf-babel-hmac] and "DTLS" to indicate support of [I-D.ietf-babel-dtls].

babel-mac-algorithms: List of supported MAC computation algorithms. Possible values include "HMAC-SHA256", "BLAKE2s-128" to indicate support for algorithms indicated in [I-D.ietf-babel-hmac].

babel-dtls-cert-types: List of supported DTLS certificate types. Possible values include "X.509" and "RawPublicKey" to indicate support for types indicated in [I-D.ietf-babel-dtls].

babel-stats-enable: Indicates whether statistics collection is enabled (true) or disabled (false) on all interfaces. When enabled, existing statistics values are not cleared and will be incremented as new packets are counted.

babel-stats-reset: An operation that resets all babel-if-stats parameters to zero. This operation has no input or output parameters.

babel-constants: A babel-constants-obj object.

babel-interfaces: A set of babel-interface-obj objects.

babel-routes: A set of babel-route-obj objects. Contains the routes known to this node.

babel-mac-key-sets: A set of babel-mac-key-set-obj objects. If this object is implemented, it provides access to parameters related to the MAC security mechanism. An implementation MAY choose to expose this object as read-only ("ro").

babel-dtls-cert-sets: A set of babel-dtls-cert-set-obj objects. If this object is implemented, it provides access to parameters related to the DTLS security mechanism. An implementation MAY choose to expose this object as read-only ("ro").

3.2. Definition of babel-constants-obj

```
object {
     uint         rw babel-udp-port;
    [ip-address   rw babel-mcast-group;]
} babel-constants-obj;
```

babel-udp-port: UDP port for sending and listening for Babel packets. Default is 6696. An implementation MAY choose to expose this parameter as read-only ("ro"). This is a 16-bit unsigned integer.

babel-mcast-group: Multicast group for sending and listening to multicast announcements on IPv6. Default is ff02::1:6. An implementation MAY choose to expose this parameter as read-only ("ro").

3.3. Definition of babel-interface-obj

```
object {
     reference            ro babel-interface-reference;
    [boolean              rw babel-interface-enable;]
     string               rw babel-interface-metric-algorithm;
    [boolean              rw babel-interface-split-horizon;]
    [uint                 ro babel-mcast-hello-seqno;]
    [uint                 ro babel-mcast-hello-interval;]
    [uint                 ro babel-update-interval;]
    [boolean              rw babel-mac-enable;]
    [reference            rw babel-if-mac-key-sets<0..*>;]
    [boolean              rw babel-mac-verify;]
    [boolean              rw babel-dtls-enable;]
    [reference            rw babel-if-dtls-cert-sets<0..*>;]
    [boolean              rw babel-dtls-cached-info;]
    [string               rw babel-dtls-cert-prefer<0..*>;]
    [boolean              rw babel-packet-log-enable;]
    [reference            ro babel-packet-log;]
    [babel-if-stats-obj   ro babel-if-stats;]
     babel-neighbor-obj   ro babel-neighbors<0..*>;
} babel-interface-obj;
```

babel-interface-reference: Reference to an interface object that can be used to send and receive IPv6 packets, as defined by the data model (e.g., YANG [RFC7950], BBF [TR-181]). Referencing syntax will be specific to the data model. If there is no set of interface objects available, this should be a string that indicates the interface name used by the underlying operating system.

babel-interface-enable: When written, it configures whether the protocol should be enabled (true) or disabled (false) on this interface. A read from the running or intended datastore indicates the configured administrative value of whether the protocol is enabled (true) or not (false). A read from the operational datastore indicates whether the protocol is actually running (true) or not (i.e., it indicates the operational state of the protocol). A data model that does not replicate parameters for running and operational datastores can implement this as two separate parameters. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-interface-metric-algorithm: Indicates the metric computation algorithm used on this interface. The value MUST be one of those listed in the babel-information-obj babel-metric-comp-algorithms parameter. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-interface-split-horizon: 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. Split horizon optimization is described in [I-D.ietf-babel-rfc6126bis], section 3.7.4. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-mcast-hello-seqno: The current sequence number in use for multicast Hellos sent on this interface. This is a 16-bit unsigned integer.

babel-mcast-hello-interval: The current interval in use for multicast Hellos sent on this interface. Units are centiseconds. This is a 16-bit unsigned integer.

babel-update-interval: The current interval in use for all updates (multicast and unicast) sent on this interface. Units are centiseconds. This is a 16-bit unsigned integer.

babel-mac-enable: Indicates whether the MAC security mechanism is enabled (true) or disabled (false). An implementation MAY choose to expose this parameter as read-only ("ro").

babel-if-mac-keys-sets: List of references to the babel-mac entries that apply to this interface. When an interface instance is created, all babel-mac-key-sets instances with babel-mac-default-apply "true" will be included in this list. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-mac-verify A Boolean flag indicating whether MACs 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. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-dtls-enable: Indicates whether the DTLS security mechanism is enabled (true) or disabled (false). An implementation MAY choose to expose this parameter as read-only ("ro").

babel-if-dtls-cert-sets: List of references to the babel-dtls-cert-sets entries that apply to this interface. When an interface instance is created, all babel-dtls-cert-sets instances with babel-dtls-default-apply "true" will be included in this list. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-dtls-cached-info: Indicates whether the cached_info extension

(see [I-D.ietf-babel-dtls] Appendix A) is included in ClientHello and ServerHello packets. The extension is included if the value is "true". An implementation MAY choose to expose this parameter as read-only ("ro").

babel-dtls-cert-prefer: List of supported certificate types, in order of preference. The values MUST be among those listed in the babel-dtls-cert-types parameter. This list is used to populate the server_certificate_type extension (see [I-D.ietf-babel-dtls] Appendix A) in a Client Hello. Values that are present in at least one instance in the babel-dtls-certs object of a referenced babel-dtls instance and that have a non-empty babel-cert-private-key will be used to populate the client_certificate_type extension in a Client Hello.

babel-packet-log-enable: Indicates whether packet logging is enabled (true) or disabled (false) on this interface.

babel-packet-log: A reference or url link to a file that contains a timestamped log of packets received and sent on babel-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 babel-packet-log-enable. Implementations will need to carefully manage and limit memory used by packet logs.

babel-if-stats: Statistics collection object for this interface.

babel-neighbors: A set of babel-neighbor-obj objects.

3.4. Definition of babel-if-stats-obj

```
object {
     uint   ro babel-sent-mcast-hello;
     uint   ro babel-sent-mcast-update;
     uint   ro babel-sent-ucast-hello;
     uint   ro babel-sent-ucast-update;
     uint   ro babel-sent-IHU;
     uint   ro babel-received-packets;
} babel-if-stats-obj;
```

babel-sent-mcast-hello: A count of the number of multicast Hello packets sent on this interface.

babel-sent-mcast-update: A count of the number of multicast update packets sent on this interface.

babel-sent-ucast-hello: A count of the number of unicast Hello

packets sent on this interface.

babel-sent-ucast-update: A count of the number of unicast update packets sent on this interface.

babel-sent-IHU: A count of the number of IHU packets sent on this interface.

babel-received-packets: A count of the number of Babel packets received on this interface.

3.5. Definition of babel-neighbor-obj

```
object {
     ip-address   ro babel-neighbor-address;
    [binary       ro babel-hello-mcast-history;]
    [binary       ro babel-hello-ucast-history;]
     uint         ro babel-txcost;
     uint         ro babel-exp-mcast-hello-seqno;
     uint         ro babel-exp-ucast-hello-seqno;
    [uint         ro babel-ucast-hello-seqno;]
    [uint         ro babel-ucast-hello-interval;]
    [uint         ro babel-rxcost;]
    [uint         ro babel-cost;]
} babel-neighbor-obj;
```

babel-neighbor-address: IPv4 or IPv6 address the neighbor sends packets from.

babel-hello-mcast-history: The multicast Hello history of whether or not the multicast Hello packets prior to babel-exp-mcast-hello-seqno were received. A binary sequence where the most recently received Hello is expressed as a "1" placed in the left-most bit, with prior bits shifted right (and "0" bits placed between prior Hello bits and most recent Hello for any not-received Hellos). This value should be displayed using hex digits ([0-9a-fA-F]). See [I-D.ietf-babel-rfc6126bis], section A.1.

babel-hello-ucast-history: The unicast Hello history of whether or not the unicast Hello packets prior to babel-exp-ucast-hello-seqno were received. A binary sequence where the most recently received Hello is expressed as a "1" placed in the left-most bit, with prior bits shifted right (and "0" bits placed between prior Hello bits and most recent Hello for any not-received Hellos). This value should be displayed using hex digits ([0-9a-fA-F]). See [I-D.ietf-babel-rfc6126bis], section A.1.

babel-txcost: Transmission cost value from the last IHU packet

received from this neighbor, or maximum value to indicate the IHU hold timer for this neighbor has expired. See [I-D.ietf-babel-rfc6126bis], section 3.4.2. This is a 16-bit unsigned integer.

babel-exp-mcast-hello-seqno: 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. This is a 16-bit unsigned integer; if the data model uses zero (0) to represent NULL values for unsigned integers, the data model MAY use a different data type that allows differentiation between zero (0) and NULL.

babel-exp-ucast-hello-seqno: 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. This is a 16-bit unsigned integer; if the data model uses zero (0) to represent NULL values for unsigned integers, the data model MAY use a different data type that allows differentiation between zero (0) and NULL.

babel-ucast-hello-seqno: The current sequence number in use for unicast Hellos sent to this neighbor. If unicast Hellos are not being sent, this MUST be NULL. This is a 16-bit unsigned integer; if the data model uses zero (0) to represent NULL values for unsigned integers, the data model MAY use a different data type that allows differentiation between zero (0) and NULL.

babel-ucast-hello-interval: The current interval in use for unicast Hellos sent to this neighbor. Units are centiseconds. This is a 16-bit unsigned integer.

babel-rxcost: 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. See [I-D.ietf-babel-rfc6126bis], section 3.4.3. This is a 16-bit unsigned integer.

babel-cost: The link cost, as 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. This is a 16-bit unsigned integer.

3.6. Definition of babel-route-obj

```
object {
     ip-address   ro babel-route-prefix;
     uint         ro babel-route-prefix-length;
     binary       ro babel-route-router-id;
     reference    ro babel-route-neighbor;
     uint         ro babel-route-received-metric;
     uint         ro babel-route-calculated-metric;
     uint         ro babel-route-seqno;
     ip-address   ro babel-route-next-hop;
     boolean      ro babel-route-feasible;
     boolean      ro babel-route-selected;
} babel-route-obj;
```

babel-route-prefix: Prefix (expressed in IP address format) for which this route is advertised.

babel-route-prefix-length: Length of the prefix for which this route is advertised.

babel-route-router-id: The router-id of the router that originated this route.

babel-route-neighbor: Reference to the babel-neighbors entry for the neighbor that advertised this route.

babel-route-received-metric: The metric with which this route was advertised by the neighbor, or maximum value to indicate the route was recently retracted and is temporarily unreachable (see Section 3.5.5 of [I-D.ietf-babel-rfc6126bis]). This metric will be NULL if the route was not received from a neighbor but was generated through other means. At least one of babel-route-calculated-metric and babel-route-received-metric MUST be non-NULL. Having both be non-NULL is expected for a route that is received and subsequently advertised. This is a 16-bit unsigned integer; if the data model uses zero (0) to represent NULL values for unsigned integers, the data model MAY use a different data type that allows differentiation between zero (0) and NULL.

babel-route-calculated-metric: A calculated metric for this route.

How the metric is calculated is implementation-specific. Maximum value indicates the route was recently retracted and is temporarily unreachable (see Section 3.5.5 of [I-D.ietf-babel-rfc6126bis]). At least one of babel-route-calculated-metric and babel-route-received-metric MUST be non-NULL. Having both be non-NULL is expected for a route that is received and subsequently advertised. This is a 16-bit unsigned integer; if the data model uses zero (0) to represent NULL values for unsigned integers, the data model MAY use a different data type that allows differentiation between zero (0) and NULL.

babel-route-seqno: The sequence number with which this route was advertised. This is a 16-bit unsigned integer.

babel-route-next-hop: The next-hop address of this route. This will be empty if this route has no next-hop address.

babel-route-feasible: A Boolean flag indicating whether this route is feasible, as defined in Section 3.5.1 of [I-D.ietf-babel-rfc6126bis]).

babel-route-selected: A Boolean flag indicating whether this route is selected (i.e., whether it is currently being used for forwarding and is being advertised).

3.7. Definition of babel-mac-key-set-obj

```
object {
     boolean             rw babel-mac-default-apply;
     babel-mac-key-obj   rw babel-mac-keys<0..*>;
} babel-mac-key-set-obj;
```

babel-mac-default-apply: A Boolean flag indicating whether this object instance is applied to all new babel-interface instances, by default. If "true", this instance is applied to new babel-interfaces instances at the time they are created, by including it in the babel-if-mac-key-sets list. If "false", this instance is not applied to new babel-interfaces instances when they are created. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-mac-keys: A set of babel-mac-key-obj objects.

3.8. Definition of babel-mac-key-obj

```
object {
     string      rw babel-mac-key-name;
     boolean     rw babel-mac-key-use-send;
     boolean     rw babel-mac-key-use-verify;
     binary      -- babel-mac-key-value;
     string      rw babel-mac-key-algorithm;
    [operation      babel-mac-key-test;]
} babel-mac-key-obj;
```

babel-mac-key-name: 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 MUST NOT be empty and can only be provided when this instance is created (i.e., it is not subsequently writable). The value MAY be auto-generated if not explicitly supplied when the instance is created.

babel-mac-key-use-send: Indicates whether this key value is used to compute a MAC and include that MAC in the sent Babel packet. A MAC for sent packets is computed using this key if the value is "true". If the value is "false", this key is not used to compute a MAC to include in sent Babel packets. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-mac-key-use-verify: 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 with the MAC in an incoming packet. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-mac-key-value: 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 when read, 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 babel-mac-key-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-128", the length MUST be between 0 and 32 bytes inclusive, as described in [RFC7693].

babel-mac-key-algorithm The name of the MAC algorithm used with this key. The value MUST be the same as one of the enumerations listed in the babel-mac-algorithms parameter. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-mac-key-test: An operation that allows the MAC key and MAC algorithm to be tested to see if they produce an expected outcome. Input to this operation are a binary string and a calculated MAC (also in the format of a binary string) for the binary string. The implementation is expected to create a MAC over the binary string using the babel-mac-key-value and the babel-mac-key-algorithm. The output of this operation is a Boolean indication that the calculated MAC matched the input MAC (true) or the MACs did not match (false).

3.9. Definition of babel-dtls-cert-set-obj

```
object {
     boolean               rw babel-dtls-default-apply;
     babel-dtls-cert-obj   rw babel-dtls-certs<0..*>;
} babel-dtls-cert-set-obj;
```

babel-dtls-default-apply: A Boolean flag indicating whether this object instance is applied to all new babel-interface instances, by default. If "true", this instance is applied to new babel-interfaces instances at the time they are created, by including it in the babel-interface-dtls-certs list. If "false", this instance is not applied to new babel-interfaces instances when they are created. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-dtls-certs: A set of babel-dtls-cert-obj objects. This contains both certificates for this implementation to present for authentication, and to accept from others. Certificates with a non-empty babel-cert-private-key can be presented by this implementation for authentication.

3.10. Definition of babel-dtls-cert-obj

```
object {
     string      rw babel-cert-name;
     string      rw babel-cert-value;
     string      rw babel-cert-type;
     binary      -- babel-cert-private-key;
} babel-dtls-cert-obj;
```

babel-cert-name: A unique name for this 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). The value MAY be auto-generated if not explicitly supplied when the instance is created.

babel-cert-value: The certificate in PEM format [RFC7468]. This value MUST be provided when this instance is created, and is not subsequently writable.

babel-cert-type: The name of the certificate type of this object instance. The value MUST be the same as one of the enumerations listed in the babel-dtls-cert-types parameter. This value can only be provided when this instance is created, and is not subsequently writable.

babel-cert-private-key: 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 when read, or through permissions, or other means. This value can only be provided when this instance is created, and is not subsequently writable.

4. Extending the Information Model

Implementations MAY extend this information model with other parameters or objects. For example, an implementation MAY choose to expose Babel route filtering rules by adding a route filtering object with parameters appropriate to how route filtering is done in that implementation. The precise means used to extend the information model would be specific to the data model the implementation uses to expose this information.

5. Security Considerations

This document defines a set of information model objects and parameters that may be exposed to be visible from other devices, and some of which may be configured. Securing access to and ensuring the integrity of this data is in scope of and the responsibility of any data model derived from this information model. Specifically, any YANG [RFC7950] data model is expected to define security exposure of the various parameters, and a [TR-181] data model will be secured by the mechanisms defined for the management protocol used to transport it.

Misconfiguration (whether unintentional or malicious) can prevent reachability or cause poor network performance (increased latency, jitter, etc.). The information in this model discloses network topology, which can be used to mount subsequent attacks on traffic traversing the network.

This information model defines objects that can allow credentials (for this device, for trusted devices, and for trusted certificate authorities) to be added and deleted. Public keys may be exposed through this model. This model requires that private keys and MAC keys never be exposed. Certificates used by [I-D.ietf-babel-dtls] implementations use separate parameters to model the public parts (including the public key) and the private key.

MAC keys are allowed to be as short as zero-length. This is useful for testing. Network operators are RECOMMENDED to follow current best practices for key length and generation of keys related to the MAC algorithm associated with the key. Short (and zero-length) keys are highly susceptible to brute force attacks and therefore SHOULD NOT be used. See the Security Considerations section of [I-D.ietf-babel-hmac] for additional considerations related to MAC keys.

This information model uses key sets and certification sets to provide a means of grouping keys and certificates. This makes it easy to use a different set per interface, the same set for one or more interfaces, have a default set in case a new interface is instantiated and to change keys and certificates as needed.

6. IANA Considerations

This document has no IANA actions.

7. Acknowledgements

Juliusz Chroboczek, Toke Hoeiland-Joergensen, David Schinazi, Antonin Decimo, Acee Lindem, and Carsten Bormann have been very helpful in refining this information model.

The language in the Notation section was mostly taken from [RFC8193].

8. References

8.1. Normative References

[I-D.ietf-babel-dtls]
Decimo, A., Schinazi, D., and J. Chroboczek, "Babel Routing Protocol over Datagram Transport Layer Security", Work in Progress, Internet-Draft, draft-ietf-babel-dtls-10, 30 June 2020, <http://www.ietf.org/internet-drafts/draft-ietf-babel-dtls-10.txt>.

[I-D.ietf-babel-hmac]
Do, C., Kolodziejak, W., and J. Chroboczek, "MAC authentication for the Babel routing protocol", Work in Progress, Internet-Draft, draft-ietf-babel-hmac-12, 4 September 2020, <http://www.ietf.org/internet-drafts/draft-ietf-babel-hmac-12.txt>.

[I-D.ietf-babel-rfc6126bis]
Chroboczek, J. and D. Schinazi, "The Babel Routing Protocol", Work in Progress, Internet-Draft, draft-ietf-babel-rfc6126bis-20, 24 August 2020, <http://www.ietf.org/internet-drafts/draft-ietf-babel-rfc6126bis-20.txt>.

[ISO.10646]
International Organization for Standardization, "Information Technology - Universal Multiple-Octet Coded Character Set (UCS)", ISO Standard 10646:2014, 2014.

[libpcap] Wireshark, "Libpcap File Format", 2015, <https://wiki.wireshark.org/Development/LibpcapFileFormat>.

[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>.

[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>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[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>.

[RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <https://www.rfc-editor.org/info/rfc7468>.

[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>.

[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>.

8.2. Informative References

[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>.

[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>.

[RFC8193] Burbridge, T., Eardley, P., Bagnulo, M., and J. Schoenwaelder, "Information Model for Large-Scale Measurement Platforms (LMAPs)", RFC 8193, DOI 10.17487/RFC8193, August 2017, <https://www.rfc-editor.org/info/rfc8193>.

[TR-181] Broadband Forum, "Device Data Model", <http://cwmp-data-models.broadband-forum.org/>.

Authors' Addresses

Barbara Stark
AT&T
Atlanta, GA,
United States of America

Email: barbara.stark@att.com

Mahesh Jethanandani
VMware
California
United States of America

Email: mjethanandani@gmail.com

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

Table of Contents

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.

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

- "ZZZZ" --> the assigned RFC value for Babel Information Model [I-D.ietf-babel-information-model]

- 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 a 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 overriden 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.";
```

```
        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 perpective:

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>.

[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>
   <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.
-->
```

```
<?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>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

Babel Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 1, 2021

M. Jethanandani
Kloud Services
B. Stark
AT&T

January 28, 2021

YANG Data Model for Babel
draft-ietf-babel-yang-model-07

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 August 1, 2021.

Copyright Notice

Table of Contents

1. Introduction

This document defines a data model for the Babel routing protocol [RFC8966]. 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.

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

- "ZZZZ" --> the assigned RFC value for Babel Information Model [I-D.ietf-babel-information-model]

- Revision date in model, in the format 2021-01-26 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 a 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-key-set* [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 overriden 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 [RFC8966].

<CODE BEGINS> file "ietf-babel@2021-01-26.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 2021-01-26 {
    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.";
    reference
      "draft-ietf-babel-hmac: MAC authentication for Babel Routing
       Protocol.";
  }

  feature dtls-supported {
    description
      "This implementation supports DTLS based security.";
    reference
      "draft-ietf-babel-dtls: Babel Routing Protocol over Datagram
       Transport Layer Security.";
  }

  feature hmac-sha256-supported {
    description
      "This implementation supports hmac-sha256 MAC algorithm.";
    reference
      "draft-ietf-babel-hmac: MAC authentication for Babel Routing
       Protocol.";
  }

  feature blake2s-supported {
    description
      "This implementation supports blake2s MAC algorithms.
       Specifically, BLAKE2-128 is supported.";
    reference
```

```
       "draft-ietf-babel-hmac: MAC authentication for Babel Routing
        Protocol.";
   }

   feature x-509-supported {
     description
       "This implementation supports x-509 certificate type.";
     reference
       "draft-ietf-babel-dtls: Babel Routing Protocol over Datagram
        Transport Layer Security.";
   }

   feature raw-public-key-supported {
     description
       "This implementation supports raw-public-key certificate type.";
     reference
       "draft-ietf-babel-dtls: Babel Routing Protocol over Datagram
        Transport Layer Security.";
   }

   /*
    * 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.";
     reference
       "draft-ietf-babel-rfc6126bis: The Babel Routing Protocol,
        Section A.2.1.";
   }

   identity etx {
     if-feature etx-supported;
     base "metric-comp-algorithms";
     description
       "Expected Transmission Count.";
     reference
       "draft-ietf-babel-rfc6126bis: The Babel Routing Protocol,
        Section A.2.2.";
   }
```

```
   /*
    * 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 algorithms supported. Specifically, BLAKE2-128 is
        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. When enabled,
           existing statistics values are not cleared and will be
           incremented as new packets are counted.";
      }

      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-key-set/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 MACs 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.";
          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.
             draft-ietf-babel-dtls: Babel Routing Protocol over
             Datagram Transport Layer Security, Appendix A.";
          }

          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
               draft-ietf-babel-dtls: Babel Routing Protocol over
               Datagram Transport Layer Security, Appendix A.";
          }

          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-set {
        key "name";

        description
          "A mac key set 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 object
             instance is applied to all new interfaces, by default.
             If 'true', this instance is applied to new babel-
             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.";
            reference
              "RFC ZZZZ: Babel Information Model, Section 3.8.";
          }

          leaf use-send {
            type boolean;
            mandatory true;
            description
              "Indicates whether this key value is used to compute a
               MAC and include that MAC in the sent Babel packet. A MAC
               for sent packets is computed using this key if the value
               is 'true'. If the value is 'false', this key is not used
               to compute a MAC to include in 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 'BLAKE2-128', 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 MAC
           algorithm to be tested to see if they produce an
           expected outcome. Input to this operation are a
           binary string and a calculated MAC (also in the
           format of a binary string) for the binary string.
           The implementation is expected to create a MAC over
           the binary string using the value and algorithm.
           The output of this operation is a binary indication that
           the calculated MAC matched the input MAC (true) or the
           MACs did not match (false).";
         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 MAC over this string using the value and
               the algorithm defined as part of the mac-key-set.";
            reference
              "RFC ZZZZ: Babel Information Model, Section 3.8.";
          }

          leaf mac {
            type binary;
            mandatory true;
            description
              "Input to this operation includes a MAC.
               The implementation is expected to calculate a MAC
               over the string using the value and algorithm of
               this key object and compare its calculated MAC to
               this input MAC.";
            reference
              "RFC ZZZZ: Babel Information Model, Section 3.8.";
          }
        }
        output {
```

```
            leaf indication {
              type boolean;
              mandatory true;
              description
                "The output of this operation is a binary indication
                 that the calculated MAC matched the input MAC (true)
                 or the MACs did not match (false).";
              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 object
             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 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 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.";
            }
          }
        }

        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 perpective:

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

[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>.

[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>.

[RFC8966] Chroboczek, J. and D. Schinazi, "The Babel Routing Protocol", RFC 8966, DOI 10.17487/RFC8966, January 2021, <https://www.rfc-editor.org/info/rfc8966>.

6.2. Informative References

[I-D.ietf-babel-information-model] Stark, B. and M. Jethanandani, "Babel Information Model", draft-ietf-babel-information-model-11 (work in progress), August 2020.

[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-key-set>
            <name>hmac-sha256</name>
            <keys>
              <name>hmac-sha256-keys</name>
              <use-send>true</use-send>
              <use-verify>true</use-verify>
              <value>base64encodedvalue==</value>
              <algorithm>hmac-sha256</algorithm>
            </keys>
          </mac-key-set>
        </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>
   <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.
-->
```

```
<?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>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