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