Babel Information Model
draft-ietf-babel-information-model-06

Abstract

   This Babel Information Model can be used to create data models under
   various data modeling regimes. It allows a Babel implementation (via
   a management protocol or interface) to report on its current state
   and may allow some limited configuration of protocol constants.

Status of This Memo

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

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

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

   This Internet-Draft will expire on January 9, 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
   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 (contingent
   on a management protocol with Babel support being implemented). 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 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
      +-- babel-supported-link-properties
      +-- self-seqno
      +-- babel-metric-comp-algorithms
      +-- babel-security-supported
      +-- babel-hmac-enable
      +-- babel-hmac-algorithms
      +-- babel-dtls-enable
      +-- 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-link-properties
      | +-- babel-interface-metric-algorithm
      | +-- babel-mcast-hello-seqno
      | +-- babel-mcast-hello-interval
      | +-- babel-update-interval
      | +-- babel-interfaces-hmac-keys
      | +-- babel-hmac-algorithm
      | +-- babel-hmac-verify
      | +-- babel-interfaces-dtls-certs
      | +-- 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-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-nbr-stats
        |  |  |  +-- babel-sent-ucast-hello
        |  |  |  +-- babel-sent-ucast-update
        |  |  |  +-- babel-sent-IHU
        |  |  |  +-- babel-received-hello
        |  |  |  +-- babel-received-update
        |  |  |  +-- babel-received-IHU
        +-- 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-hmac
        |  +-- babel-hmac-default-apply
        |  |  +-- babel-hmac-key-name
        |  |  +-- babel-hmac-key-use-sign
        |  |  +-- babel-hmac-key-use-verify
        |  |  +-- babel-hmac-key-value
        +-- babel-dtls
        |  +-- babel-dtls-default-apply
        |  |  +-- 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:

o enable/disable Babel

o create/delete babel-hmac objects

o create/delete babel-dtls objects

o enable/disable statistics collection

o Constant: UDP port

o Constant: IPv6 multicast group

o Interface: Link type

o Interface: enable/disable Babel on this interface

o Interface: sets of HMAC keys

o Interface: HMAC algorithm

o Interface: verify received HMAC packets

o Interface: set of DTLS certificates

o Interface: use cached info extensions

o Interface: preferred order of certificate types

o Interface: enable/disable packet log

o HMAC-keys: create/delete entries

o HMAC-keys: use to sign packets

o HMAC-keys: use to verify packets

o DTLS-certs: create/delete entries

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

o HMAC key values

o 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;
     string               ro babel-supported-link-properties<1..*>;
    [uint                 ro babel-self-seqno;]
     string               ro babel-metric-comp-algorithms<1..*>;
     string               ro babel-security-supported<0..*>;
    [boolean              ro babel-hmac-enable;]
    [string               ro babel-hmac-algorithms<1..*>;]
    [boolean              ro babel-dtls-enable;]
    [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-hmac-obj       rw babel-hmac<0..*>;]
    [babel-dtls-obj       rw babel-dtls<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.

babel-supported-link-properties: Lists the collections of link
   properties supported by this instance of Babel. Valid enumeration
   values are defined in the Babel Link Properties registry (see
   [Section 6](#)).

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 "k-out-of-j", and "ETX".

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

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

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

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-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, including
   neighbor-specific statistics (babel-nbr-stats).

babel-stats-reset: An operation that resets all babel-if-stats and
   babel-nbr-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-hmac: A babel-hmac-obj object. If this object is implemented,
   it provides access to parameters related to the HMAC security
   mechanism. An implementation MAY choose to expose this object as
   read-only ("ro").

babel-dtls: A babel-dtls-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 ff02:0:0:0:0:0:1:6.
   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-link-properties;
      string              ro babel-interface-metric-algorithm;
     [uint                ro babel-mcast-hello-seqno;]
     [uint                ro babel-mcast-hello-interval;]
     [uint                ro babel-update-interval;]
     [reference           rw babel-interface-hmac-keys<0..*>;]
     [string              rw babel-hmac-algorithm;]
     [boolean             rw babel-hmac-verify;]
     [reference           rw babel-interface-dtls-certs<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 IPv6 interface object 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-link-properties: Indicates the properties of the link. The
value MUST be one of those listed in the babel-supported-link-
properties parameter. Valid enumeration values are identified in
Babel Link Properties registry. 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.

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

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

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

babel-interface-dtls-certs: List of references to the babel-dtls
   entries that apply to this interface. When an interface instance
   is created, all babel-dtls 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-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-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-nbr-stats-obj ro babel-nbr-stats;]
      } 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 0. This is a
16-bit unsigned integer.

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 0. This is a 16-bit unsigned integer.

babel-ucast-hello-seqno: The current sequence number in use for
unicast Hellos sent to this neighbor. This is a 16-bit unsigned
integer.

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: 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. This is a 16-bit unsigned integer.

babel-nbr-stats: Statistics collection object for this neighbor.

3.6. Definition of babel-nbr-stats-obj

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

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

babel-sent-ucast-update: A count of the number of unicast update
   packets sent to this neighbor.

babel-sent-IHU: A count of the number of IHU packets sent to this
   neighbor.

babel-received-hello: A count of the number of Hello packets
   received from this neighbor.

babel-received-update: A count of the number of update packets
   received from this neighbor.

babel-received-IHU: A count of the number of IHU packets received
   from this neighbor.

3.7. 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: router-id of the source router for which this
   route is advertised.

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 0 (zero) 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-
zero. Having both be non-zero is expected for a route that is
received and subsequently advertised. This is a 16-bit unsigned
integer.

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-
zero. Having both be non-zero is expected for a route that is
received and subsequently advertised. This is a 16-bit unsigned
integer.

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).

3.8. Definition of babel-hmac-obj

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

babel-hmac-default-apply: A Boolean flag indicating whether this
babel-hmac instance is applied to all 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 babel-
interface-hmac-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-hmac-keys: A set of babel-hmac-keys-obj objects.

3.9. Definition of babel-hmac-keys-obj

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

   babel-hmac-key-name: A unique name for this HMAC 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.

   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 HMAC is computed from this key for comparing an incoming
      packet. An implementation MAY choose to expose this parameter as
      read-only ("ro").

   babel-key-value: The value of the HMAC 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 can only be provided when this instance is created, and
      is not subsequently writable.

   babel-hmac-test: An operation that allows the HMAC 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-hmac-key-
      value and the babel-hmac-algorithm. The output of this operation
      is the resulting hash, as a binary string.

3.10. Definition of babel-dtls-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 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 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.

3.11. Definition of babel-dtls-certs-obj

```
   object {
        string                 ro babel-cert-name;
        string                 ro babel-cert-value;
        string                 ro 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
      can only be provided when this instance is created, and is not
      subsequently writable.

   babel-cert-value: The DTLS certificate in PEM format [RFC7468].
      This value can only 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, 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.

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 and shared secrets
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]) are expected to define what credentials can be
used with those mechanisms.

6. IANA Considerations

This document defines a Babel Link Properties registry for the values
of the babel-link-properties and babel-supported-link-properties
parameters to be listed under the Babel Routing Protocol registry.

Valid Babel Link Properties names are normatively defined as

o MUST be at least 1 character and no more than 20 characters long

o MUST contain only US-ASCII [RFC0020] letters 'A' - 'Z' and 'a' -
   'z', digits '0' - '9', and hyphens ('-', ASCII 0x2D or decimal 45)

o MUST contain at least one letter ('A' - 'Z' or 'a' - 'z')

o MUST NOT begin or end with a hyphen

o hyphens MUST NOT be adjacent to other hyphens

The rules for Link Properties names, excepting the limit of 20
characters maximum, are also expressed below (as a non-normative
convenience) using ABNF [RFC5234].

```
        SRVNAME = *(1*DIGIT [HYPHEN]) ALPHA *([HYPHEN] ALNUM)
        ALNUM   = ALPHA / DIGIT      ; A-Z, a-z, 0-9
        HYPHEN  = %x2D               ; "-"
        ALPHA   = %x41-5A / %x61-7A ; A-Z / a-z [RFC5234]
        DIGIT   = %x30-39            ; 0-9        [RFC5234]
```

The allocation policy of this registry is Specification Required
[RFC8126].

The initial values in the "Babel Link Properties" registry are:

```
+----------+----------------------------------------+--------------+
| Name     | Describes Links with Properties        | Reference    |
+----------+----------------------------------------+--------------+
| other    | no link property information available | (this        |
|          |                                        | document)    |
| tunnel   | a tunneled interface over unknown      | (this        |
|          | physical link                          | document)    |
| wired    |                                        | (this        |
|          |                                        | document)    |
| wireless |                                        | (this        |
|          |                                        | document)    |
| exp-*    | Reserved for Experimental Use          | (this        |
|          |                                        | document)    |
+----------+----------------------------------------+--------------+
```

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-11 (work in
          progress), June 2019.

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

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80,
          RFC 20, DOI 10.17487/RFC0020, October 1969,
          <https://www.rfc-editor.org/info/rfc20>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for
          Writing an IANA Considerations Section in RFCs", BCP 26,
          RFC 8126, DOI 10.17487/RFC8126, June 2017,
          <https://www.rfc-editor.org/info/rfc8126>.

[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-07 (work in progress), July 2019.

[I-D.ietf-babel-hmac]
          Do, C., Kolodziejak, W., and J. Chroboczek, "HMAC
          authentication for the Babel routing protocol", draft-
          ietf-babel-hmac-08 (work in progress), July 2019.

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

   [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>.

   [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax
              Specifications: ABNF", STD 68, RFC 5234,
              DOI 10.17487/RFC5234, January 2008,
              <https://www.rfc-editor.org/info/rfc5234>.

   [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/>.

Appendix A. Open Issues

   All open issues have been closed.

   Closed Issues:

   1.   See minutes of IETF 104 for discussion of issues that led to
        changes noted for 2019-07-08

   2.   HMAC spec adds other parameters to neighbor table. Check these
        to see if any need to be readable or writable. / None were
        identified.

   3.   Actions to add and delete HMAC and DTLS credentials, and
        parameters that allow credential to be identified without
        allowing access to private credential info. Will have separate

sub-tables for HMAC and DTLS credentials. / Instead, there is a normative statement that the parameter values must never be supplied when read.

4.   Consider the following statistics: under interface object: sent multicast Hello, sent updates, received Babel messages; under neighbor object: sent unicast Hello, sent updates, sent IHU, received Hello, received updates, received IHUs. Would also need to enable/disable stats and clear stats.

5.   Message log (optional to implement) is still in. Support for the libpcap file format is "SHOULD".

6.   Single security table with (optional) reference to interfaces that security mechanism applies to. / This actually became separate objects for DTLS and HMAC.

7.   Should ABNF be normative in IANA Considerations section? Decision was to leave it as is.

8.   I want to get rid of the security log, because all Babel messages (which should be defined as all messages to/from the udp-port) are be logged by message-log. I don't like message log as it is. I think if logging is enabled it should just write to a text file. This will mean there also needs to be a means of downloading/reading the log file. Closed by having single log for all messages to/from udp port and log is represented by a string that can be reference to filename or some other part of the overall data model (depends on data model).

9.   Check description of enable parameters to make sure ok for YANG and TR-181. Closed by updating description to be useful for YANG and TR-181, using language consistent with YANG descriptions. Done.

10.   Distinguish signed and unsigned integers? All integers are unsigned and size is mentioned in description of each uint parameter.

11.   Datatype of the router-id: Closed by introducing binary datatype and using that for router-id

12.   babel-neighbor-address as IPv6-only: Closed by leaving as is (IPv4 and IPv6)

13.   babel-implementation-version includes the name of the implementation: Closed by adding "name" to description

14.   Delete external-cost?: Closed by deleting.

15.   Would it be useful to define some parameters for reporting
      statistics or logs? [2 logs are now included. If others are
      needed they need to be proposed. See Open Issues for additional
      thoughts on logs and statistics.]

16.   Closed by defining base64 type and using it for all router IDs:
      "babel-self-router-id: Should this be an opaque 64-bit value
      instead of int?"

17.   Closed as "No": Do we need a registry for the supported security
      mechanisms? [Given the current limited set, and unlikelihood of
      massive expansion, I don't think so. But we can if someone
      wants it.]

18.   This draft must be reviewed against draft-ietf-babel-rfc6126bis.
      [I feel like this has been adequately done, but I could be
      wrong.]

19.   babel-interfaces-obj: Juliusz:"This needs further discussion, I
      fear some of these are implementation details." [In the absence
      of discussion, the current model stands. Note that all but
      link-type and the neighbors sub-object are optional. If an
      implementation does not have any of the optional elements then
      it simply doesn't have them and that's fine.]

20.   Would it be useful to define some parameters specifically for
      security anomalies? [The 2 logs should be useful in identifying
      security anomalies. If more is needed, someone needs to
      propose.]

21.   I created a basic security model. It's useful for single (or
      no) active security mechanism (e.g., just HMAC, just DTLS, or
      neither); but not multiple active (both HMAC and DTLS -- which
      is not the same as HMAC of DTLS and would just mean that HMAC
      would be used on all unencrypted messages -- but right now the
      model doesn't allow for configuring HMAC of unencrypted messages
      for routers without DTLS, while DTLS is used if possible). OK?
      [No-one said otherwise.]

22.   babel-external-cost may need more work. [if no comment, it will
      be left as is]

23.   babel-hello-[mu]cast-history: the Hello history is formated as
      16 bits, per A.1 of 6126bis. Is that a too implementation
      specific? [We also now have an optional-to-implement log of

received messages, and I made these optional. So maybe this is
ok?]

24. rxcost, txcost, cost: is it ok to model as integers, since
     6126bis 2.1 says costs and metrics need not be integers. [I
     have them as integers unless someone insists on something else.]

25. For the security log, should it also log whether the credentials
     were considered ok? [Right now it doesn't and I think that's ok
     because if you log Hellos it was ok and if you don't it wasn't.]

26. Should Babel link types have an IANA registry? [Agreed to do
     this at IETF 102.]

[Appendix B](#). Change Log

Individual Drafts:

v00 2016-07-07 EBD: Initial individual draft version

v01 2017-03-13: Addressed comments received in 2016-07-15 email from
    J. Chroboczek

Working group drafts:

v00 2017-07-03: Addressed points noted with "oops" in
    https://www.ietf.org/proceedings/98/slides/slides-98-babel-babel-information-model-00.pdf

v01 2018-01-02: Removed item from issue list that was agreed (in
    Prague) not to be an issue. Added description of data types under
    Notation section, and used these in all data types. Added babel-
    security and babel-trust.

v02 2018-04-05:

   * changed babel-version description to babel-implementation-
      version

   * replace optional babel-interface-seqno with optional babel-
      mcast-hello-seqno and babel-ucast-hello-seqno

   * replace optional babel-interface-hello-interval with optional
      babel-mcast-hello-interval and babel-ucast-hello-interval

   * remove babel-request-trigger-ack

* remove "babel-router-id: router-id of the neighbor"; note that
  parameter had previously been removed but description had
  accidentally not been removed

* added an optional "babel-cost" field to babel-neighbors object,
  since the spec does not define how exactly the cost is computed
  from rxcost/txcost

* deleted babel-source-garbage-collection-time

* change babel-lossy-link to babel-link-type and make this an
  enumeration; added at top level babel-supported-link-types so
  which are supported by this implementation can be reported

* changes to babel-security-obj to allow self credentials to be
  one or more instances of a credential object. Allowed trusted
  credentials to include CA credentials; made some parameter name
  changes

* updated references and Introduction

* added Overview section

* deleted babel-sources-obj

* added feasible Boolean to routes

* added section to briefly describe extending the information
  model.

* deleted babel-route-neighbor

* tried to make definition of babel-interface-reference clearer

* added security and message logs

v03 2018-05-31:

* added reference to RFC 8174 (update to RFC 2119 on key words)

* applied edits to Introduction text per Juliusz email of
  2018-04-06

* Deleted sentence in definition of "int" data type that said it
  was also used for enumerations. Changed all enumerations to
  strings. The only enumerations were for link types, which are
  now "ethernet", "wireless", "tunnel", and "other".

   * deleted [ip-address babel-mcast-group-ipv4;]

   * babel-external-cost description changed

   * babel-security-self-cred: Added "any private key component of a
      credential MUST NOT be readable;"

   * hello-history parameters put recent Hello in most significant
      bit and length of parameter is not constrained.

   * babel-hello-seqno in neighbors-obj changed to babel-exp-mcast-
      hello-seqno and babel-exp-ucast-hello-seqno

   * added babel-route-neighbor back again. It was mistakenly
      deleted

   * changed babel-route-metric and babel-route-announced-metric to
      babel-route-received-metric and babel-route-calculated-metric

   * changed model of security object to put list of supported
      mechanisms at top level and separate security object per
      mechanism. This caused some other changes to the security
      object

   v04 2018-10-15:

   * changed babel-mcast-group-ipv6 to babel-mcast-group

   * link type parameters changed to point to newly defined registry

   * babel-ucast-hello-interval moved to neighbor object

   * babel-ucast-hello-seqno moved to neighbor object

   * babel-neighbor-ihu-interval deleted

   * in log descriptions, included statement that there SHOULD be
      ability to clear logs

   * added IANA registry for link types

   * added "ro" and "rw" to tables for read-write and read-only

   * added metric computation parameter to interface

   v05 2019-01-15:

  * security modeled with single table under information-obj and
    references to interfaces that instance applies to

  * changed int to uint because all integers in model were
    unsigned; added size of integer to description of each uint
    parameter

  * deleted log object and made single message log that points to
    file or other data model object used to maintain logs

  * deleted babel-credentials; there are no more "common" objects;
    hmac keys and DTLS certificates are more explicitly modeled

  * changed definition of babel-security-supported

  * added parameters for HMAC and DTLS

  * added statistics

  * changed all instances of "message" to "packet"

v06 2019-07-08:

  * changed Link Type registry in IANA considerations to Lik
    Property Types

  * changed direction of reference for HMAC and DTLS objects to be
    from interface to these objects

  * provided DTLS certificate objects with a unique name

  * changed received and calculated metric descriptions to make
    clear that it is ok to have both

  * constrained interface reference to only IPv6 interfaces

Authors' Addresses

Barbara Stark
AT&T
Atlanta, GA
US

Email: barbara.stark@att.com

Mahesh Jethanandani
VMware
California
US

Email: mjethanandani@gmail.com