Stream: Internet Engineering Task Force (IETF)
RFC: 9046
Category: Informational
Published: June 2021
ISSN: 2070-1721
Authors: B. Stark *AT&T* M. Jethanandani *Kloud Services*

RFC 9046
Babel Information Model

Abstract

The 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 document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9046.

Copyright Notice

Table of Contents

1. Introduction

Babel is a loop-avoiding, distance-vector routing protocol defined in [RFC8966]. [RFC8967] defines a security mechanism that allows Babel packets to be cryptographically authenticated, and [RFC8968] 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 the 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 [RFC8966] 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 BCP 14 [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 [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. The following is a descriptive list of the parameters that are not required to be read-only:

- enable/disable Babel
- create/delete Babel Message Authentication Code (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. [RFC8966] describes this as an arbitrary string of 8 octets.

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", as described in [RFC8966], Appendix A.2.1, and "ETX", as described in [RFC8966], Appendix A.2.2.

babel-security-supported: List of supported security mechanisms. Possible values include "MAC" to indicate support of [RFC8967] and "DTLS" to indicate support of [RFC8968].

babel-mac-algorithms: List of supported MAC computation algorithms. Possible values include "HMAC-SHA256" and "BLAKE2s-128" to indicate support for algorithms indicated in [RFC8967].

babel-dtls-cert-types: List of supported certificate types. Possible values include "X.509" and "RawPublicKey" to indicate support for types indicated in [RFC8968].

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] and Broadband Forum (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-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 [RFC8966], 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-key-sets: List of references to the babel-mac-key-sets 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 [RFC8968], 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 [RFC8968], Appendix A) in a ClientHello. 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 ClientHello.

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 a .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 "I Heard You" (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 leftmost 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 [RFC8966], Appendix 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 leftmost bit, with prior bits shifted right (and "0" bits placed between prior Hello bits and the most recent Hello for any not-received Hellos). This value should be displayed using hex digits ([0-9a-fA-F]). See [RFC8966], Appendix A.1.

babel-txcost: Transmission cost value from the last IHU packet received from this neighbor, or the maximum value to indicate the IHU hold timer for this neighbor has expired. See [RFC8966], 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 [RFC8966], 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 the maximum value to indicate the route was recently retracted and is temporarily unreachable (see Section 3.5.4 of [RFC8966]). This metric will be NULL if the route was not received from a neighbor but was generated through other means. At least one of the following **MUST** be non-NULL: babel-route-calculated-metric or babel-route-received-metric. 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. The maximum value indicates the route was recently retracted and is temporarily unreachable (see Section 3.5.4 of [RFC8966]). At least one of the following **MUST** be non-NULL: babel-route-calculated-metric or babel-route-received-metric. 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 [RFC8966]).

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-interfaces 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 comparison 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, through permissions, or by 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 Hashed Message Authentication Code (HMAC) construction [RFC2104], the length **MUST** be between 0 and an upper limit that is at least the size of the output length (where the "HMAC-SHA256" output length is 32 octets as described in [RFC4868]). Longer lengths **MAY** be supported but are not necessary if the management system has the ability to generate a suitably random value (e.g., by randomly generating a value or by using a key derivation technique as recommended in the security considerations in Section 7 of [RFC8967]). If the algorithm is "BLAKE2s-128", the length **MUST** be between 0 and 32 bytes inclusive as specified by [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 is 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-interfaces 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 those 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 Privacy-Enhanced Mail (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, through permissions, or by 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 and visible from other devices. Some of these information model objects and parameters 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.). Misconfiguration of security credentials can cause a denial-of-service condition for the Babel routing protocol. 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 [RFC8968] 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. It is **RECOMMENDED** that network operators 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 as described in Section 7 of [RFC8967] for additional considerations related to MAC keys; note that there are some specific key value recommendations in the fifth paragraph. It says that if it is necessary to derive keys from a human-readable passphrase, "only the derived keys should be communicated to the routers" and "the original passphrase itself should be kept on the host used to perform the key generation" (which would be the management system in the case of a remote management protocol). It also recommends that keys "should have a length of 32 octets (both for HMAC-SHA256 and BLAKE2s), and be chosen randomly".

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, use the same set for one or more interfaces, have a default set in case a new interface is instantiated, and change keys and certificates as needed.

6. IANA Considerations

This document has no IANA actions.

7. References

7.1. Normative References

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

[libpcap] GitLab, "Libpcap File Format", Wireshark Foundation, November 2020, <https://gitlab.com/wireshark/wireshark/-/wikis/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>.

[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>.

[RFC8967] Dô, C., Kolodziejak, W., and J. Chroboczek, "MAC Authentication for the Babel Routing Protocol", RFC 8967, DOI 10.17487/RFC8967, January 2021, <https://www.rfc-editor.org/info/rfc8967>.

[RFC8968] Décimo, A., Schinazi, D., and J. Chroboczek, "Babel Routing Protocol over Datagram Transport Layer Security", RFC 8968, DOI 10.17487/RFC8968, January 2021, <https://www.rfc-editor.org/info/rfc8968>.

7.2. 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", Issue: 2 Amendment 14, November 2020, <http://cwmp-data-models.broadband-forum.org/>.

Acknowledgements

Juliusz Chroboczek, Toke Høiland-Jørgensen, David Schinazi, Antonin Décimo, Roman Danyliw, Benjamin Kaduk, Valery Smyslov, Alvaro Retana, Donald Eastlake, Martin Vigoureux, 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].

Authors' Addresses

Barbara Stark
AT&T
TX
United States of America
Email: barbara.stark@att.com

Mahesh Jethanandani
Kloud Services
CA
United States of America
Email: mjethanandani@gmail.com