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 February 4, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
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
   --- self-seqno
   --- babel-metric-comp-algorithms
   --- babel-security-supported
   --- babel-hmac-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-hmac-enable
      --- babel-if-hmac-key-sets
      --- babel-hmac-algorithm
      --- babel-hmac-verify
      --- babel-dtls-enable
      --- babel-if-dtls-cert-sets
      --- babel-dtls-cached-info
      --- babel-dtls-cert-prefer
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-hmac objects
- create/delete babel-dtls objects
- enable/disable statistics collection
- Constant: UDP port
- Constant: IPv6 multicast group
- Interface: Link type
- Interface: enable/disable Babel on this interface
- Interface: sets of HMAC keys
- Interface: HMAC algorithm
- Interface: verify received HMAC packets
- Interface: set of DTLS certificates
- Interface: use cached info extensions
- Interface: preferred order of certificate types
- Interface: enable/disable packet log
- HMAC-keys: create/delete entries
- HMAC-keys: use to sign packets
- HMAC-keys: use to verify packets
- DTLS-certs: create/delete entries

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

- HMAC key values
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-hmac-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-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. 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".

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

babel-hmac-algorithms: List of supported HMAC 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, 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::0:0:0:0:1::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-mcast-groups<0..*>;
    [reference            rw babel-if-hmac-key-sets<0..*>;]
    [string               rw babel-hmac-algorithm;]
    [boolean              rw babel-hmac-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-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-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.

**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-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-if-hmac-keys-sets**: List of references to the babel-hmac entries that apply to this interface. When an interface instance is created, all babel-hmac-key-sets 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
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-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

```plaintext
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; 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-
zero. Having both be non-zero is expected for a route that is
received and subsequently advertised. This is a 16-bit unsigned
integer; but it may be represented by a data model as a signed
integer for schemas that use 0 (zero) to represent NULL with
unsigned integers and use negative numbers to represent NULL with
signed integers.

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-key-sets-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 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-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").

b babel-hmac-keys: A set of babel-hmac-keys-obj objects.

3.9. Definition of babel-hmac-keys-obj

object {
    string                    rw 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 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 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 MUST 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 MUST be a non-empty 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-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.

3.11. 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, 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 MUST be a non-empty 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. 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].

7. References

7.1. Normative References

[I-D.ietf-babel-rfc6126bis]


7.2. Informative References

[I-D.ietf-babel-dtls]

[I-D.ietf-babel-hmac]

[ISO.10646]

[RFC3339]

[RFC5234]

[RFC6241]

[RFC7950]

[RFC8193]

[TR-181]
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


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 formatted 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:

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

v07 2019-07-22:
* babel-dtls-enable and babel-hmac-enable moved to interfaces and made rw
* renamed babel-dtls and babel-hmac to babel-dtls-cert-sets and babel-hmac-key-sets and references to them from interfaces are babel-if-dtls-cert-sets and babel-if-hmac-key-sets
* https://github.com/bhstark2/babel-information-model/issues/16 with nits
* https://github.com/bhstark2/babel-information-model/issues/14 addressing parameters not allowed to be empty/null
* https://github.com/bhstark2/babel-information-model/issues/18 on IANA link properties table

v08 2019-08-04:
* Deleted IANA Considerations section
* Deleted babel-supported-link-properties and babel-link-properties in all places
* Made babel-interface-metric-algorithm rw
* Added boolean rw babel-interface-split-horizon parameter
* Replaced the "k-out-of-j" enumeration for expression of algorithmic capabilities with "2-out-of-3"
* Calculated and received metrics datatype can be signed int if needed to represent NULL value

Authors’ Addresses

Barbara Stark
AT&T
Atlanta, GA
US

Email: barbara.stark@att.com
Delay-based Metric Extension for the Babel Routing Protocol
draft-ietf-babel-rtt-extension-00

Abstract

This document defines an extension to the Babel routing protocol that uses symmetric delay in metric computation and therefore makes it possible to prefer lower latency links to higher latency ones.

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 October 28, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

The Babel routing protocol [BABEL] does not mandate a specific algorithm for computing metrics; existing implementations use a packet-loss based metric on wireless links and a simple hop-count metric on all other types of links. While this strategy works reasonably well in many networks, it fails to select reasonable routes in some topologies involving tunnels or VPNs.

Consider for example the following topology, with three routers A, B and D located in Paris and a fourth router located in Tokyo, connected through tunnels in a diamond topology.

```
+------------+
| A (Paris)  +---------------+
+------------+                |
    /        /                           /
   /        /                             /
  /        /                               /
+------------+                                     +------------+
| B (Paris) |                                     | C (Tokyo) |
+------------+                                     +------------+
    /        /                           /
   /        /                             /
  /        /                               /
+------------+                                     +------------+
| D (Paris)  +---------------+
+------------+
```
When routing traffic from A to D, it is obviously preferable to use the local route through B, as this is likely to provide better service quality and lower monetary cost than the distant route through C. However, the existing implementations of Babel consider both routes as having the same metric, and will therefore route the traffic through C in roughly half the cases.

In this document, we specify an extension to the Babel routing protocol that enables precise measurement of the round-trip time (RTT) of a link, and allows its usage in metric computation. Since this causes a negative feedback loop, special care is needed to ensure that the resulting network is reasonably stable (Section 2.3).

We believe that this protocol may be useful in other situations than the one described above, such as when running Babel in a congested wireless mesh network or over a complex link layer that performs its own routing; the high granularity of the timestamps used (1ms) should make it easier to experiment with RTT-based metrics on this kind of link layers.

2. Protocol operation

The protocol estimates the RTT to each neighbour (Section 2.1) which it then uses for metric computation (Section 2.2).

2.1. Delay estimation

The RTT to a neighbour is estimated using an algorithm due to Mills [MILLS], originally developed for the HELLO routing protocol and later used in NTP [NTP].

A Babel speaker periodically sends a multicast Hello message over all of its interfaces (Section 3.4.1 of [BABEL]). This Hello is usually accompanied with a set of IHU messages, one per neighbour (Section 3.4.2 of [BABEL]).

In order to enable the computation of RTTs, a node A SHOULD include in every Hello that it sends a timestamp $t_1$ (according to A’s clock). When a node B receives A’s Hello, it records in its neighbour table the timestamp $t_1$ as well as the time $t_1'$ according to its own (B’s) clock at which it received the packet.

When B later sends an IHU to A, it SHOULD attach to the IHU the timestamps $t_1$ and $t_1'$ which it has stored in its neighbour table. Additionally, it SHOULD ensure that the packet within which the IHU is sent contains a Hello TLV with an associated timestamp $t_2'$ (according to B’s clock). Symmetrically, A will record in its neighbour table the timestamp $t_2'$ as well as the time $t_2$ (according
to A’s clock) at which it has received the Hello. This is illustrated in the following sequence diagram:

```
A          B
|      |
|      |
|      +
|      |
|      |
|      + t1'
|      |
|      + t2'
|      /|
|      / |
|      /  |
|      /   |
|      /    |
|      /     |
|      /      |
|      /       |
|      /        |
|      /         |
|      /          |
|      /           |
|      + t1''      |
|      |
+ t2' +
```

A then estimates the RTT between A and B as \((t2 - t1) - (t2' - t1')\).

This algorithm has a number of desirable properties. First, since there is no requirement that \(t1'\) and \(t2'\) be equal, the protocol remains asynchronous -- the only change to Babel’s message scheduling is the requirement that a packet containing an IHU also contains a Hello. Second, since only differences of timestamps according to a single clock are computed, it does not require synchronised clocks. Third, it requires very little additional state -- a node only needs to store the two timestamps associated with the last hello received from each neighbour. Finally, since it only requires piggybacking one or two timestamps on each Hello and IHU packet, it makes efficient use of network resources.

In principle, this algorithm is inaccurate in the presence of clock drift (i.e. when A’s and B’s clocks are running at different frequencies). However, \(t2' - t1'\) is usually on the order of seconds, and significant clock drift is unlikely to happen at that time scale.
2.2. Metric computation

The algorithm described in the previous section allows computing an RTT to every neighbour. How to map this value to a link cost is a local implementation matter.

Obviously, the mapping should be monotonic (larger RTTs imply larger costs). In addition, in order to enhance stability (Section 2.3), the mapping should be bounded -- above a certain RTT, all links are equally bad.

2.2.1. Example metric computation

The current implementation of Babel uses the following function for mapping RTTs to link costs, parameterised by three parameters rtt-min, rtt-max and max-rtt-penalty:

\[
\text{cost} \begin{cases} 
C + \text{max-rtt-penalty} & \text{for } 0 \leq \text{RTT} < \text{rtt-min} \\
C + \frac{\text{max-rtt-penalty} \times (\text{RTT} - \text{rtt-min})}{\text{rtt-max} - \text{rtt-min}} & \text{for } \text{rtt-min} \leq \text{RTT} < \text{rtt-max} \\
C + \text{max-rtt-penalty} & \text{for } \text{RTT} \geq \text{rtt-max}
\end{cases}
\]

For RTTs below rtt-min, the link cost is just the nominal cost of a single hop, C. Between rtt-min and rtt-max, the cost increases linearly; above rtt-max, the constant value max-rtt-penalty is added to the nominal cost.
2.3. Stability issues

Using delay as an input to the routing metric in congested networks gives rise to a negative feedback loop: low RTT encourages traffic, which in turn causes the RTT to increase. In a congested network, such a feedback loop can cause persistent oscillations.

The current implementation of Babel uses three techniques that collaborate to limit the frequency of oscillations:

- the measured RTT is smoothed, which limits Babel’s response to short-term RTT variations;
- the mapping function is bounded, which avoids switching between congested routes;
- a hysteresis algorithm is applied to the metric before route selection, which limits the worst-case frequency of route oscillations.

These techniques are discussed in more detail in [DELAY-BASED].

2.4. Backwards and forwards compatibility

This protocol extension stores the data that it requires within sub-TLVs of Babel’s Hello and IHU TLVs. As discussed in Section 4 of [BABEL-EXT], implementations that do not understand this extension will silently ignore the sub-TLVs while parsing the rest of the TLVs that they contain. In effect, this extension supports building hybrid networks consisting of extended and unextended routers, and while such networks might suffer from sub-optimal routing, they will not suffer from blackholes or routing loops.

If a sub-TLV defined in this extension is longer than expected, the additional data is silently ignored. This provision is made in order to allow a future version of this document to extend the packet format with additional data.

3. Packet format

This extension defines the Timestamp sub-TLV [BABEL-EXT], whose Type field has value 3. This sub-TLV can be contained within a Hello sub-TLV, in which case it carries a single timestamp, or within an IHU sub-TLV, in which case it carries two timestamps.

Timestamps are encoded as 32-bit unsigned integers, expressed in units of one microsecond, counting from an arbitrary origin.
Timestamps wrap around every 4295 seconds, or slightly more than one hour.

3.1. Timestamp sub-TLV in Hello TLVs

When contained within a Hello TLV, the Timestamp sub-TLV has the following format:

0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 3    |    Length     |      Transmit timestamp       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Fields:

Type    Set to 3 to indicate a Timestamp sub-TLV.
Length  The length of the body, exclusive of the Type and Length fields.
Transmit timestamp  The time at which the packet containing this sub-TLV was sent, according to the sender’s clock.

If the Length field is larger than the expected 4 octets, the sub-TLV MUST be processed normally and any extra data contained in this sub-TLV MUST be silently ignored.

3.2. Timestamp sub-TLV in IHU TLVs

When contained in an IHU TLV destined for node A, the Timestamp sub-TLV has the following format:

0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 3    |    Length     |        Origin timestamp       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |        Receive timestamp      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Fields:

Type    Set to 3 to indicate a Timestamp sub-TLV.
Length    The length of the body, exclusive of the Type and Length fields.

Origin timestamp  A copy of the transmit timestamp of the last Timestamp sub-TLV contained in a Hello TLV received from node A.

Receive timestamp  The time at which the last Hello with a Timestamp sub-TLV was received from node A according to the sender’s clock.

If the Length field is larger than the expected 8 octets, the sub-TLV MUST be processed normally and any extra data contained in this sub-TLV MUST be silently ignored.

4. IANA Considerations

IANA is instructed to add the following entry to the "Babel Sub-TLV Types" registry:

+------+-----------+-----------------+
| Type | Name      | Reference       |
|------|-----------+-----------------|
| 3    | Timestamp | (this document) |
+------+-----------+-----------------+

5. Security Considerations

This extension merely adds additional timestamping data to two of the TLVs sent by a Babel router, and does not significantly change the security properties of the Babel protocol.

6. References

6.1. Normative References


6.2. Informative References


Authors’ Addresses

Baptiste Jonglez
ENS Lyon
France

Email: baptiste.jonglez@ens-lyon.org

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

Email: jch@irif.fr
Abstract

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

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 23, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document.
1. Introduction

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

1.1. Note to RFC Editor

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

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

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

1.3. Tree Diagram Annotations

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

2. Babel Module

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

2.1. Information Model

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

2.2. Tree Diagram

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

augment /rt:routing/rt:control-plane-protocols
/rt:control-plane-protocol:
  +++rw babel!
    +++ro version? string
    +++rw enable boolean
    +++ro router-id binary
    +++ro link-properties* identityref
    +++ro sequence-number? uint16
    +++ro metric-comp-algorithms* identityref
    +++ro security-supported* identityref
    +++ro hmac-algorithms* identityref
    +++ro dtls-cert-types* identityref
    +++rw stats-enable? boolean
    +++rw constants
    | ...
    +++rw interfaces* [reference]
    | ...
    +++rw hmac* [name]
    | ...
    +++rw dtls* [name]
  ...

augment /rt:routing/rt:ribs/rt:rib/rt:routes/rt:route:
  +++ro routes* [prefix]
    +++ro prefix inet:ip-prefix
    +++ro router-id? binary
    +++ro neighbor? leafref
    +++ro received-metric? uint16
    +++ro calculated-metric? uint16
    +++ro seqno? uint16
    +++ro next-hop? inet:ip-address
    +++ro feasible? boolean
    +++ro selected? boolean

The interfaces subtree describes attributes such as interface object that is being referenced, the type of link as enumerated by Babel Link Properties, and whether the interface is enabled or not.

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

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

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

2.3. YANG Module

This module augments a YANG Data Model for Interface Management [RFC8343], YANG Routing Management [RFC8349], and imports definitions from Common YANG Data Types [RFC6991].

<CODE BEGINS> file "ietf-babel@2019-07-22.yang"

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

  import ietf-yang-types {
    prefix yt;
    reference
    "RFC 6991 - Common YANG Data Types.";
  }
  import ietf-inet-types {
    prefix inet;
    reference
    "RFC 6991 - Common YANG Data Types.";
  }
  import ietf-interfaces {
    prefix if;
    reference
    "RFC 8343 - A YANG Data Model for Interface Management";
  }
  import ietf-routing {
    prefix "rt";
    reference
    "RFC 8349 - YANG Routing Management";
  }

  organization
  "IETF Babel routing protocol Working Group";

  contact
  "WG Web: http://tools.ietf.org/wg/babel/
  WG List: babel@ietf.org"
This YANG module defines a model for the Babel routing protocol.

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved. Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.

revision 2019-07-22 {
  description
    "Initial version.";
  reference
    "RFC XXX: Babel YANG Data Model.";
}

/*
 * Identities
 */
identity link-property {
  description
    "Base identity from which all Babel Link Types are derived.";
}

identity other {
  base "link-property";
  description
    "No link property information available.";
}

identity tunnel {
  base "link-property";
  description
    "A tunneled interface over unknown physical link.";
}
identity wired {
    base "link-property";
    description
        "A wired link with fixed physical properties.";
}

identity wireless {
    base "link-property";
    description
        "Wireless link type for Babel Routing Protocol.";
}

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

identity k-out-of-j {
    base "metric-comp-algorithms";
    description
        "k-out-of-j algorithm.";
}

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

/*
 * Babel security type identities
 */

identity security-supported {
    description
        "Base identity from which all Babel security types are derived.";
}

identity hmac {
    base security-supported;
    description
        "HMAC supported.";
}

identity dtls {
    base security-supported;
    description
        "Datagram Transport Layer Security (DTLS) supported.";
    reference
        ...
}
"RFC 6347, Datagram Transport Layer Security Version 1.2.";
}

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

identity hmac-sha256 {
  base hmac-algorithms;
  description
    "HMAC-SHA256 algorithm supported.";
}

identity blake2s {
  base hmac-algorithms;
  description
    "BLAKE2s algorithm supported.";
  reference
    "RFC 7693, The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC).";
}

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

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

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

/*
 * Babel routing protocol identity.

Jethanandani & Stark Expires January 23, 2020 [Page 8]
identity babel {
  base "rt:routing-protocol";
  description
    "Babel routing protocol";
}

/*
 * Features
 */

/*
 * Features supported
 */

/*
 * Typedefs
 */

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

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

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

    leaf neighbor {
      type leafref {
        path "/rt:routing/rt:control-plane-protocols/" +
leaf reference {
    type uint16;
    description
    "Reference to the babel-neighbors entry for the neighbor
    that advertised this route.";
    reference
    "RFC ZZZZ, Babel Information Model, Section 3.6.";
}

leaf seqno {
    type uint16;
    description
    "The sequence number with which this route was advertised.";
    reference
    "RFC ZZZZ, Babel Information Model, Section 3.6.";
}
leaf next-hop {
  type inet:ip-address;
  description
    "The next-hop address of this route. This will be empty if
    this route has no next-hop address.";
  reference
    "RFC ZZZZ, Babel Information Model, Section 3.6.";
}

leaf feasible {
  type boolean;
  description
    "A boolean flag indicating whether this route is feasible.";
  reference
    "RFC ZZZZ, Babel Information Model, Section 3.6,
    draft-ietf-babel-rfc6126bis, The Babel Routing Protocol,
    Section 3.5.1.";
}

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

description
  "A set of babel-route-obj objects. Includes received and
  routes routes.";
reference
  "RFC ZZZZ, Babel Information Model, Section 3.1.";

description
  "Common grouping for routing used in RIB augmentation.";
}

/*
 * Data model
 */

augment "/rt:routing/rt:control-plane-protocols/" +
  "rt:control-plane-protocol" {
  when "derived-from-or-self(rt:type, 'babel')" {
    description
      "Augmentation is valid only when the instance of routing type
      is of type 'babel'.";
  }
description
"Augment the routing module to support features such as VRF.";
reference
"YANG Routing Management, RFC 8349, Lhotka & Lindem, March 2018."

container babel {
  presence "A Babel container."
  leaf version {
    type string;
    config false;
    description
    "The name and version of this implementation of the Babel protocol.";
    reference
    "RFC ZZZZ, Babel Information Model, Section 3.1.";
  }
  leaf enable {
    type boolean;
    mandatory true;
    description
    "When written, it configures whether the protocol should be enabled. A read from the <running> or <intended> datastore therefore indicates the configured administrative value of whether the protocol is enabled or not.

    A read from the <operational> datastore indicates whether the protocol is actually running or not, i.e. it indicates the operational state of the protocol.";
    reference
    "RFC ZZZZ, Babel Information Model, Section 3.1.";
  }
  leaf router-id {
    type binary;
    config false;
    mandatory "true";
    description
    "Every Babel speaker is assigned a router-id, which is an arbitrary string of 8 octets that is assumed to be unique across the routing domain";
    reference
    "RFC ZZZZ, Babel Information Model, Section 3.1, rfc6126bis, The Babel Routing Protocol. Section 3.";
  }
}
leaf-list link-properties {
    type identityref {
        base link-property;
    }
    config false;
    min-elements 1;
    description
        "Lists the collections of link properties supported by this
         instance of Babel. Valid enumeration values are defined
         in the Babel Link Properties registry."
    reference
        "RFC ZZZZ, Babel Information Model, Section 3.1."
}

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

leaf-list metric-comp-algorithms {
    type identityref {
        base "metric-comp-algorithms"
    }
    config false;
    min-elements 1;
    description
        "List of cost compute algorithms supported by this
         implementation of Babel."
    reference
        "RFC ZZZZ, Babel Information Model, Section 3.1."
}

leaf-list security-supported {
    type identityref {
        base "security-supported"
    }
    config false;
    min-elements 1;
    description
        "Babel security mechanism used by this implementation or
         per interface."
    reference
        "RFC ZZZZ, Babel Information Model, Section 3.1."
}
leaf-list hmac-algorithms {
  type identityref {
    base hmac-algorithms;
  }
  config false;
  description
    "List of supported HMAC computation algorithms. Possible
     values include ‘HMAC-SHA256’, ‘BLAKE2s’.";
  reference
    "RFC ZZZZ, Babel Information Model, Section 3.1.";
}

leaf-list dtls-cert-types {
  type identityref {
    base dtls-cert-types;
  }
  config false;
  description
    "List of supported DTLS certificate types. Possible values
     include ‘X.509’ and ‘RawPublicKey’.";
  reference
    "RFC ZZZZ, Babel Information Model, Section 3.1.";
}

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

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

  leaf mcast-group {
    type inet:ip-address;
    default "ff02:0:0:0:0:0:1:6";
  }
}
description
  "Multicast group for sending and receiving multicast
  announcements on IPv6."
reference
  "RFC ZZZZ, Babel Information Model, Section 3.2."
}
description
  "Babel Constants object."
reference
  "RFC ZZZZ, Babel Information Model, Section 3.1."
}
list interfaces {
  key "reference";
leaf reference {
  type if:interface-ref;
description
    "Reference to an interface object as defined by the data
    model (e.g., YANG, BBF TR-181); data model is assumed to
    allow for referencing of interface objects which may be at
    any layer (physical, Ethernet MAC, IP, tunneled IP, etc.).
    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."
reference
    "RFC ZZZZ, Babel Information Model, Section 3.3."
}
leaf enable {
  type boolean;
default "true";
description
  "If true, babel sends and receives messages on this
  interface. If false, babel messages received on this
  interface are ignored and none are sent."
reference
  "RFC ZZZZ, Babel Information Model, Section 3.3."
}
leaf link-properties {
  type identityref {
    base link-property;
  }
default "wired";
description
  "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 identity-refs derived from properties identified in Babel Link Properties registry.

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

leaf metric-algorithm {
  type identityref {
    base metric-comp-algorithms;
  }
  default "k-out-of-j";
  description
  "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."
}

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

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

leaf update-interval {
  type uint16;
  units centiseconds;
  config false;
  description
  "The current update interval in use for this interface. Units are centiseconds.";
  reference

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

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

leaf hmac-algorithm {
  type identityref {
    base hmac-algorithms;
  }
  description
      "The name of the HMAC algorithm used on this interface. The value is one of the identities listed as part of babel-hmac-algorithms at a global level.";
  reference
      "RFC ZZZZ, Babel Information Model, Section 3.3.";
}

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

leaf-list dtls-certs {
  type leafref {
    path "../../dtls/name";
  }
  description
    "List of references to the babel-dtls-cert-sets 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.";
  reference
    "RFC ZZZZ, Babel Information Model, Section 3.3.";
}

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

leaf-list dtls-cert-prefer {
  type leafref {
    path "../../dtls/certs/type";
  }
  ordered-by user;
  description
    "List of supported certificate types, in order of
    preference. The values MUST be among those listed in 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.";
}
leaf packet-log-enable {
  type boolean;
  description
    "If true, logging of babel packets received on this interface is enabled; if false, babel packets are not logged."
  reference
    "RFC ZZZZ, Babel Information Model, Section 3.3.";
}

leaf packet-log {
  type inet:uri;
  config false;
  description
    "A reference or url link to a file that contains a timestamped log of packets received and sent on 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 packet-log-enable."
  reference
    "RFC ZZZZ, Babel Information Model, Section 3.3.";
}

container stats {
  config false;
  leaf sent-mcast-hello {
    type yt:counter32;
    description
      "A count of the number of multicast Hello packets sent on this interface."
    reference
      "RFC ZZZZ, Babel Information Model, Section 3.4.";
  }
  leaf sent-mcast-update {
    type yt:counter32;
    description
      "A count of the number of multicast update packets sent on this interface."
    reference
      "RFC ZZZZ, Babel Information Model, Section 3.4.";
  }
}
leaf received-packets {
  type yt:counter32;
  description
    "A count of the number of Babel packets received on
     this interface.";
  reference
    "RFC ZZZZ, Babel Information Model, Section 3.4.";
}
action reset {
  input {
    leaf reset-at {
      type yt:date-and-time;
      description
        "The time when the reset was issued.";
    }
  }
  output {
    leaf reset-finished-at {
      type yt:date-and-time;
      description
        "The time when the reset finished.";
    }
  }
}

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  leaf received-hello {
    type yt:counter32;
    description
    "A count of the number of Hello packets received from
this neighbor.";
    reference
    "RFC ZZZZ, Babel Information Model, Section 3.6.";
  }
}
leaf received-update {
    type yt:counter32;
    description
    "A count of the number of update packets received
    from this neighbor.";
    reference
    "RFC ZZZZ, Babel Information Model, Section 3.6.";
}

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

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

description
"Statistics collection object for this neighbor.";
reference
"RFC ZZZZ, Babel Information Model, Section 3.6.";
}

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

description
"A set of Babel Interface objects.";
reference
"RFC ZZZZ, Babel Information Model, Section 3.3.";
list hmac {
  key "name";

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

  leaf default-apply {
      type boolean;
      description
      "A Boolean flag indicating whether this babel-hmac instance is applied to all new interfaces, by default. If 'true', this instance is applied to new babel-interfaces instances at the time they are created, by including it in the babel-interface-hmac-keys list. If 'false', this instance is not applied to new babel-interfaces instances when they are created.";
      reference
      "RFC ZZZZ, Babel Information Model, Section 3.8.";
  }
}

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

  leaf name {
      type string;
      mandatory "true";
      description
      "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.";
      reference
      "RFC ZZZZ, Babel Information Model, Section 3.9.";
  }

  leaf use-sign {
      type boolean;
      mandatory "true";
      description
      "Indicates whether this key value is used to sign sent Babel packets. Sent packets are signed using this key
if the value is ‘true’. If the value is ‘false’, this key is not used to sign sent Babel packets.
reference "RFC ZZZZ, Babel Information Model, Section 3.9.";
}
leaf use-verify {
    type boolean;
    mandatory "true";
    description "Indicates whether this key value is used to verify incoming Babel packets. This key is used to verify incoming packets if the value is ‘true’. If the value is ‘false’, no HMAC is computed from this key for comparing an incoming packet.";
    reference "RFC ZZZZ, Babel Information Model, Section 3.9.";
}
leaf value {
    type binary;
    mandatory "true";
    description "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 MUST be provided when this instance is created, and is not subsequently writable.";
    reference "RFC ZZZZ, Babel Information Model, Section 3.9.";
}
action test {
    input {
        leaf test-string {
            type binary;
            mandatory "true";
            description "The test string on which this test has to be performed.";
        }
    }
    output {
        leaf resulting-hash {
            type binary;
            mandatory "true";
            description
        }
    }
}
"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.";
reference
"RFC ZZZZ, Babel Information Model, Section 3.9.";
}
}
description
"A set of babel-hmac-keys-obj objects.";
reference
"RFC ZZZZ, Babel Information Model, Section 3.8.";
}
description
"A babel-hmac-obj object. If this object is implemented, it provides access to parameters related to the HMAC security mechanism.";
reference
"RFC ZZZZ, Babel Information Model, Section 3.1.";
}
list dtls {
  key "name";

  leaf name {
    type string;
    description
    "TODO: This attribute does not exist in the model, but is needed for this model to work.";
  }

  leaf default-apply {
    type boolean;
    mandatory "true";
    description
    "A Boolean flag indicating whether this babel-dtls instance is applied to all new interfaces, by default. If 'true', this instance is applied to new babel-interfaces instances at the time they are created, by including it in the babel-interface-dtls-certs list. If 'false', this instance is not applied to new babel-interfaces instances when they are created.";
    reference
"
list certs {
    key "name";
    min-elements "1";

    leaf name {
        type string;
        description "A unique name that identifies the cert in the list.";
    }

    leaf value {
        type string;
        mandatory "true";
        description "The DTLS certificate in PEM format [RFC7468]. This value can only be provided when this instance is created, and is not subsequently writable.";
        reference "RFC ZZZZ, Babel Information Model, Section 3.11.";
    }

    leaf type {
        type identityref {
            base dtls-cert-types;
        }
        mandatory "true";
        description "The name of the certificate type of this object instance. The value MUST be the same as one of the enumerations listed in the babel-dtls-cert-types parameter. This value can only be provided when this instance is created, and is not subsequently writable.";
        reference "RFC ZZZZ, Babel Information Model, Section 3.11.";
    }

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

reference
"RFC ZZZZ, Babel Information Model, Section 3.11.";

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

description
"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.";

reference
"RFC ZZZZ, Babel Information Model, Section 3.10.";

description
"A babel-dtls-obj object. If this object is implemented, it provides access to parameters related to the DTLS security mechanism.";

reference
"RFC ZZZZZ, Babel Information Model, Section 3.1";

description
"Babel Information Objects.";
reference
"RFC ZZZZ, Babel Information Model, Section 3."
)
)
augment "/rt:routing/rt:ribs/rt:rib/rt:routes/rt:route" {
when "derived-from(rt:source-protocol, 'babel')" {
description
"Augmentation is valid for a routes whose source protocol is Babel."
}
description
"Babel specific route attributes.";
uses routes;
}
}
<CODE ENDS>

2.4. Example

The following snippet demonstrates how this data module can be configured. In this example, the routing protocol being configured is Babel, and statistics gathering is enabled.

<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<control-plane-protocols>
<control-plane-protocol>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</config>
3. IANA Considerations

This document registers one URIs and one YANG module.

3.1. URI Registrations


3.2. YANG Module Name Registration

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

Name: ietf-babel
prefix: babel
reference: RFC XXXX

4. Security Considerations

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

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

There are a number of data nodes defined in the YANG module which are writable/created/deleted (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., <edit-config>) to these data nodes without proper protection can have a negative effect on network operations.

These are the subtrees and data nodes and their sensitivity/vulnerability:

5. Acknowledgements

6. References
6.1.  Normative References

[I-D.ietf-babel-rfc6126bis]


6.2.  Informative References

[I-D.ietf-babel-information-model]


Appendix A.  An Appendix

Authors’ Addresses

Mahesh Jethanandani
VMware
California
USA

Email: mjethanandani@gmail.com