

Babel routing protocol
Internet-Draft
Intended status: Informational
Expires: April 25, 2019

B. Stark
AT&T
October 22, 2018

Babel Information Model
draft-ietf-babel-information-model-04

Abstract

This Babel Information Model can be used to create data models under various data modeling regimes (e.g., YANG). It allows a Babel implementation (via a management protocol such as NETCONF) 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 April 25, 2019.

Copyright Notice

Copyright (c) 2018 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.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 1.1. Requirements Language | 3 |
| 1.2. Notation | 3 |
| 2. Overview | 4 |
| 3. The Information Model | 5 |
| 3.1. Definition of babel-information-obj | 5 |
| 3.2. Definition of babel-constants-obj | 6 |
| 3.3. Definition of babel-interfaces-obj | 7 |
| 3.4. Definition of babel-neighbors-obj | 8 |
| 3.5. Definition of babel-security-obj | 10 |
| 3.6. Definition of babel-routes-obj | 11 |
| 4. Common Objects | 12 |
| 4.1. Definition of babel-credential-obj | 12 |
| 4.2. Definition of babel-log-obj | 13 |
| 5. Extending the Information Model | 13 |
| 6. Security Considerations | 13 |
| 7. IANA Considerations | 14 |
| 8. Acknowledgements | 15 |
| 9. References | 15 |
| 9.1. Normative References | 15 |
| 9.2. Informative References | 15 |
| Appendix A. Open Issues | 17 |
| Appendix B. Change Log | 19 |
| Author's Address | 21 |

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 messages to be cryptographically authenticated, and [I-D.ietf-babel-dtls] defines a security mechanism that allows Babel messages 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. |
| credentials | An opaque type representing credentials needed by a cryptographic mechanism to secure communication. Data models must expand this opaque type as needed and required by the security protocols utilized. |
| datetime | A type representing a date and time using the Gregorian calendar. The datetime format MUST conform to RFC 3339 [RFC3339]. |
| int | A type representing signed or unsigned integer numbers. This information model does not define a precision nor |

does it make a distinction between signed and unsigned number ranges.

- ip-address A type representing an IP address. This type supports both IPv4 and IPv6 addresses.
- string A type representing a human-readable string consisting of a (possibly restricted) subset of Unicode and ISO/IEC 10646 [ISO.10646] characters.
- uri A type representing a Uniform Resource Identifier as defined in STD 66 [RFC3986].

2. Overview

The Information Model is hierarchically structured as follows:

information object
 includes implementation version, router id, this node seqno,
 enable flag parameters, supported security mechanisms
constants object (exactly one per information object)
 includes UDP port and optional multicast group parameters
interfaces object
 includes interface reference, Hello seqno and intervals,
 update interval, link type, metric computation parameters
neighbors object
 includes neighbor IP address, Hello history, cost parameters
security object (per interface)
 includes enable flag, self credentials (credential object), trusted credentials (credential object)
security object (common to all interfaces)
 includes enable flag, self credentials (credential object), trusted credentials (credential object)
routes object
 includes route prefix, source router, reference to advertising neighbor, metric, sequence number, whether route is feasible, whether route is selected

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

- o enable/disable Babel
- o Constant: UDP port
- o Constant: IPv6 multicast group

- o Interface: Link type
- o Interface: External cost (must be configurable if implemented, but implementation is optional)
- o Interface: enable/disable Babel on this interface
- o Interface: enable/disable message log
- o Security: enable/disable this security mechanism
- o Security: self credentials
- o Security: trusted credentials
- o Security: enable/disable security log

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

3. The Information Model

3.1. Definition of babel-information-obj

```
object {  
    string          ro babel-implementation-version;  
    boolean         rw babel-enable;  
    binary          ro babel-self-router-id;  
    string          ro babel-supported-link-types<1..*>;  
    [int           ro babel-self-seqno;]  
    string          ro babel-metric-comp-algorithms<1..*>;  
    string          ro babel-security-supported<0..*>;  
    babel-constants-obj ro babel-constants;  
    babel-interfaces-obj ro babel-interfaces<0..*>;  
    babel-routes-obj ro babel-routes<0..*>;  
    babel-security-obj ro babel-security<0..*>;  
} babel-information-obj;
```

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

babel-enable: When written, it configures whether the protocol should be enabled (true) or disabled (false). A read from the running or intended datastore indicates the configured administrative value of whether the protocol is enabled (true) or not (false). A read from the operational datastore indicates whether the protocol is

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

babel-self-router-id: The router-id used by this instance of the Babel protocol to identify itself. [I-D.ietf-babel-rfc6126bis] describes this as an arbitrary string of 8 octets.

babel-supported-link-types: Lists the set of link types supported by this instance of Babel. Valid enumeration values are defined in the Babel Link Types registry (see Section 7).

babel-self-seqno: The current sequence number included in route updates for routes originated by this node.

babel-metric-comp-algorithms: List of supported cost computation algorithms. Possible values include "k-out-of-j", and "ETX".

babel-security-supported: List of supported security mechanisms. As Babel security mechanisms are defined, they will need to indicate what enumeration value is to be used to represent them in this parameter.

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

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

babel-security: A babel-security-obj object that applies to all interfaces. If this object is implemented, it allows a security mechanism to be enabled or disabled in a manner that applies to all Babel messages on all interfaces.

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

3.2. Definition of babel-constants-obj

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

babel-udp-port: UDP port for sending and listening for Babel messages. Default is 6696. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-mcast-group: Multicast group for sending and listening to multicast announcements on IPv6. Default is ff02:0:0:0:0:0:1:6. An implementation MAY choose to expose this parameter as read-only ("ro").

3.3. Definition of babel-interfaces-obj

```
object {  
    string                ro babel-interface-reference;  
    [boolean              rw babel-interface-enable;]  
    int                   rw babel-link-type;  
    string                ro babel-interface-metric-algorithm;  
    [int                  ro babel-mcast-hello-seqno;]  
    [int                  ro babel-mcast-hello-interval;]  
    [int                  ro babel-update-interval;]  
    [boolean              rw babel-message-log-enable;]  
    [babel-log-obj        ro babel-message-log<0..*>;]  
    babel-neighbors-obj   ro babel-neighbors<0..*>;  
    [babel-security-obj   ro babel-interface-security<0..*>;]  
} babel-interfaces-obj;
```

babel-interface-reference: Reference to an interface object as defined by the data model (e.g., YANG [RFC7950], 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.

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

babel-link-type: Indicates the type of link. Valid enumeration values are identified in Babel Link Types registry. An implementation MAY choose to expose this parameter as read-only ("ro").

babel-interface-metric-algorithm: Indicates the metric computation algorithm used on this interface. The value **MUST** be one of those listed in the babel-information-obj babel-metric-comp-algorithms parameter.

babel-mcast-hello-seqno: The current sequence number in use for multicast hellos sent on this interface.

babel-mcast-hello-interval: The current interval in use for multicast hellos sent on this interface.

babel-update-interval: The current interval in use for all updates (multicast and unicast) sent on this interface.

babel-message-log-enable: When written, it configures whether logging should be enabled (true) or disabled (false). A read from the running or intended datastore indicates the configured administrative value of whether logging is enabled (true) or not (false). A read from the operational datastore indicates whether logging is actually running (true) or not (i.e., it indicates the operational state). 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-message-log: Log entries that have timestamp of a received Babel message and the entire received Babel message (including Ethernet frame and IP headers, if possible). An implementation must restrict the size of this log, but how and what size is implementation-specific. If this log is implemented, a mechanism to clear it **SHOULD** be provided.

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

babel-interface-security: A babel-security-obj object that applies to this interface. If implemented, this allows security to be enabled only on specific interfaces or allows different security mechanisms to be enabled on different interfaces.

3.4. Definition of babel-neighbors-obj


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

babel-neighbor-address: IPv4 or IPv6 address the neighbor sends messages from

babel-hello-mcast-history: The multicast Hello history of whether or not the multicast Hello messages 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 messages 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 (infinity) to indicate the IHU hold timer for this neighbor has expired. See [I-D.ietf-babel-rfc6126bis], section 3.4.2.

babel-exp-mcast-hello-seqno: Expected multicast Hello sequence number of next Hello to be received from this neighbor. If multicast Hello messages are not expected, or processing of multicast messages is not enabled, this MUST be 0.

babel-exp-ucast-hello-seqno: Expected unicast Hello sequence number of next Hello to be received from this neighbor. If unicast Hello messages are not expected, or processing of unicast messages is not enabled, this MUST be 0.

babel-ucast-hello-seqno: The current sequence number in use for unicast hellos sent to this neighbor.

babel-ucast-hello-interval: The current interval in use for unicast hellos sent to this neighbor.

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.

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.

3.5. Definition of babel-security-obj

```
object {  
    string                ro babel-security-mechanism  
    boolean               rw babel-security-enable;  
    babel-credential-obj  ro babel-security-self-cred<0..*>;  
    babel-credential-obj  ro babel-security-trust<0..*>;  
    [boolean              rw babel-credvalid-log-enable;]  
    [babel-log-obj        ro babel-credvalid-log<0..*>;]  
} babel-security-obj;
```

babel-security-mechanism: The name of the security mechanism this object instance is about. The value **MUST** be the same as one of the enumerations listed in the babel-security-supported parameter.

babel-security-enable: When written, it configures whether this security mechanism should be enabled (true) or disabled (false). A read from the running or intended datastore indicates the configured administrative value of whether this security mechanism is enabled (true) or not (false). A read from the operational datastore indicates whether this security mechanism is actually running (true) or not (i.e., it indicates the operational state). 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-security-self-cred: Credentials this router presents to participate in the enabled security mechanism. Any private key component of a credential **MUST NOT** be readable. Adding and deleting credentials **MAY** be allowed.

babel-security-trust: A set of `babel-credential-obj` objects that identify the credentials of routers whose Babel messages may be trusted or of a certificate authority (CA) whose signing of a router's credentials implies the router credentials can be trusted, in the context of this security mechanism. How a security mechanism interacts with this list is determined by the mechanism. A security algorithm may do additional validation of credentials, such as checking validity dates or revocation lists, so presence in this list may not be sufficient to determine trust. Adding and deleting credentials MAY be allowed.

babel-credvalid-log-enable: When written, it configures whether logging should be enabled (true) or disabled (false). A read from the running or intended datastore indicates the configured administrative value of whether logging is enabled (true) or not (false). A read from the operational datastore indicates whether logging is actually running (true) or not (i.e., it indicates the operational state). 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-credvalid-log: Log entries that have the timestamp a message containing credentials used for peer authentication (e.g., DTLS Server Hello) was received on a Babel port, and the entire received message (including Ethernet frame and IP headers, if possible). An implementation must restrict the size of this log, but how and what size is implementation-specific. If this log is implemented, a mechanism to clear it SHOULD be provided.

3.6. Definition of `babel-routes-obj`

```
object {  
    ip-address      ro babel-route-prefix;  
    int             ro babel-route-prefix-length;  
    binary          ro babel-route-router-id;  
    string          ro babel-route-neighbor;  
    [int            ro babel-route-received-metric;  
    [int            ro babel-route-calculated-metric;  
    int             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 (infinity) 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. Either **babel-route-calculated-metric** or **babel-route-received-metric** MUST be provided.

babel-route-calculated-metric: A calculated metric for this route. How the metric is calculated is implementation-specific. Maximum value (infinity) indicates the route was recently retracted and is temporarily unreachable (see Section 3.5.5 of [I-D.ietf-babel-rfc6126bis]). Either **babel-route-calculated-metric** or **babel-route-received-metric** MUST be provided.

babel-route-seqno: The sequence number with which this route was advertised.

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).

4. Common Objects

4.1. Definition of babel-credential-obj

```
object {  
    credentials          ro babel-cred;  
} babel-credential-obj;
```

babel-cred: A credential, such as an X.509 certificate, a public key, etc. used for signing and/or encrypting Babel messages.

4.2. Definition of babel-log-obj

```
object {  
    datetime          ro babel-log-time;  
    string            ro babel-log-entry;  
} babel-log-obj;
```

babel-log-time: The date and time (according to the device internal clock setting, which may be a time relative to boot time, acquired from NTP, configured by the user, etc.) when this log entry was created.

babel-log-entry: The logged message, as a string of utf-8 encoded hex characters.

5. 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.

6. 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.

7. IANA Considerations

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

Valid Babel Link Type names are normatively defined as

- o MUST be at least 1 character and no more than 20 characters long
- o MUST contain only US-ASCII [RFC0020] letters 'A' - 'Z' and 'a' - 'z', digits '0' - '9', and hyphens ('-', ASCII 0x2D or decimal 45)
- o MUST contain at least one letter ('A' - 'Z' or 'a' - 'z')
- o MUST NOT begin or end with a hyphen
- o hyphens MUST NOT be adjacent to other hyphens

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

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

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

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

| Name | Used for Links Defined By | Reference |
|----------|--|-----------------|
| ethernet | [IEEE-802.3-2018] | (this document) |
| other | to be used when no link type information available | (this document) |
| tunnel | to be used for a tunneled interface over unknown physical link | (this document) |
| wireless | [IEEE-802.11-2016] | (this document) |
| exp-* | Reserved for Experimental Use | (this document) |

8. Acknowledgements

Juliusz Chroboczek, Toke Hoeiland-Joergensen, David Schinazi, Mahesh Jethanandani, 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].

9. References

9.1. Normative References

- [I-D.ietf-babel-rfc6126bis]
Chroboczek, J. and D. Schinazi, "The Babel Routing Protocol", draft-ietf-babel-rfc6126bis-05 (work in progress), May 2018.
- [RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <<https://www.rfc-editor.org/info/rfc20>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

9.2. Informative References

- [I-D.ietf-babel-dtls]
Decimo, A., Schinazi, D., and J. Chroboczek, "Babel Routing Protocol over Datagram Transport Layer Security", draft-ietf-babel-dtls-01 (work in progress), October 2018.
- [I-D.ietf-babel-hmac]
Do, C., Kolodziejak, W., and J. Chroboczek, "Babel Cryptographic Authentication", draft-ietf-babel-hmac-00 (work in progress), August 2018.

- [IEEE-802.11-2016]
"IEEE Standard 802.11-2016 - IEEE Standard for Information Technology - Telecommunications and information exchange between systems Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.".
- [IEEE-802.3-2018]
"IEEE Standard 802.3-2018 - IEEE Approved Draft Standard for Ethernet.".
- [ISO.10646]
International Organization for Standardization,
"Information Technology - Universal Multiple-Octet Coded Character Set (UCS)", ISO Standard 10646:2014, 2014.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8193] Burbidge, T., Eardley, P., Bagnulo, M., and J. Schoenwaelder, "Information Model for Large-Scale Measurement Platforms (LMAPs)", RFC 8193, DOI 10.17487/RFC8193, August 2017, <<https://www.rfc-editor.org/info/rfc8193>>.
- [TR-181] Broadband Forum, "Device Data Model", <<http://cwmp-data-models.broadband-forum.org/>>.

Appendix A. Open Issues

1. 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.
2. 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.
3. Security section needs further review
4. Commands to add and delete credentials, and parameters that allow credential to be identified without allowing access to private credential info
5. 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.
6. Distinguish signed and unsigned integers?
7. Review new IANA Considerations section. Should ABNF be normative?

Closed Issues:

1. Datatype of the router-id: Closed by introducing binary datatype and using that for router-id
2. babel-neighbor-address as IPv6-only: Closed by leaving as is (IPv4 and IPv6)
3. babel-implementation-version includes the name of the implementation: Closed by adding "name" to description
4. Delete external-cost?: Closed by deleting.
5. 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.]

6. 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?"
7. 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.]
8. This draft must be reviewed against draft-ietf-babel-rfc6126bis. [I feel like this has been adequately done, but I could be wrong.]
9. 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.]
10. 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.]
11. 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.]
12. babel-external-cost may need more work. [if no comment, it will be left as is]
13. 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?]
14. 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.]

15. 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.]
16. Should Babel link types have an IANA registry? [Agreed to do this at IETF 102.]

Appendix B. Change Log

Individual Drafts:

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

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

Working group drafts:

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

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

v02 2018-04-05:

- * changed babel-version description to babel-implementation-version
- * replace optional babel-interface-seqno with optional babel-mcast-hello-seqno and babel-ucast-hello-seqno
- * replace optional babel-interface-hello-interval with optional babel-mcast-hello-interval and babel-ucast-hello-interval
- * remove babel-request-trigger-ack
- * remove "babel-router-id: router-id of the neighbor"; note that parameter had previously been removed but description had accidentally not been removed
- * added an optional "babel-cost" field to babel-neighbors object, since the spec does not define how exactly the cost is computed from rxcost/txcost

- * deleted babel-source-garbage-collection-time
- * change babel-lossy-link to babel-link-type and make this an enumeration; added at top level babel-supported-link-types so which are supported by this implementation can be reported
- * changes to babel-security-obj to allow self credentials to be one or more instances of a credential object. Allowed trusted credentials to include CA credentials; made some parameter name changes
- * updated references and Introduction
- * added Overview section
- * deleted babel-sources-obj
- * added feasible Boolean to routes
- * added section to briefly describe extending the information model.
- * deleted babel-route-neighbor
- * tried to make definition of babel-interface-reference clearer
- * added security and message logs

v03 2018-05-31:

- * added reference to RFC 8174 (update to RFC 2119 on key words)
- * applied edits to Introduction text per Juliusz email of 2018-04-06
- * Deleted sentence in definition of "int" data type that said it was also used for enumerations. Changed all enumerations to strings. The only enumerations were for link types, which are now "ethernet", "wireless", "tunnel", and "other".
- * deleted [ip-address babel-mcast-group-ipv4;]
- * babel-external-cost description changed
- * babel-security-self-cred: Added "any private key component of a credential MUST NOT be readable;"

- * hello-history parameters put recent Hello in most significant bit and length of parameter is not constrained.
- * babel-hello-seqno in neighbors-obj changed to babel-exp-mcast-hello-seqno and babel-exp-ucast-hello-seqno
- * added babel-route-neighbor back again. It was mistakenly deleted
- * changed babel-route-metric and babel-route-announced-metric to babel-route-received-metric and babel-route-calculated-metric
- * changed model of security object to put list of supported mechanisms at top level and separate security object per mechanism. This caused some other changes to the security object

v04 2018-10-15:

- * changed babel-mcast-group-ipv6 to babel-mcast-group
- * link type parameters changed to point to newly defined registry
- * babel-ucast-hello-interval moved to neighbor object
- * babel-ucast-hello-seqno moved to neighbor object
- * babel-neighbor-ihu-interval deleted
- * in log descriptions, included statement that there SHOULD be ability to clear logs
- * added IANA registry for link types
- * added "ro" and "rw" to tables for read-write and read-only
- * added metric computation parameter to interface

Author's Address

Barbara Stark
AT&T
Atlanta, GA
US

Email: barbara.stark@att.com