Network Working Group J. Chroboczek
Internet-Draft PPS, University of Paris-Diderot
Intended status: Experimental June 30, 2013
Expires: January 1, 2014

Extension Mechanism for the Babel Routing Protocol
draft-chroboczek-babel-extension-mechanism-00

Abstract

This document defines the encoding of extensions to the Babel routing protocol [BABEL].

Table of Contents

1. Extending the Babel routing protocol

A Babel packet [BABEL] consists of a sequence of TLVs, each of which is a sequence of octets having an explicit type and length. The base Babel protocol has the following provisions for including extension data:

- a Babel packet with a version number different from 2 MUST be silently ignored ([BABEL], Section 4.2);

- any unknown TLV MUST be silently ignored ([BABEL], Section 4.3);

- all TLVs are self-terminating, and any extra data included in a TLV MUST be silently ignored ([BABEL], Section 4.2);

- the Flags field of the Update TLV contains 6 undefined bits that MUST be silently ignored ([BABEL], Section 4.4.9);

- any data following the last TLV of a Babel packet MUST be silently ignored ([BABEL], Section 4.2).

Each of these provisions provides a place to store data needed by extensions of the Babel protocol. However, in the absence of any further rules, different extensions of the Babel protocol might make conflicting uses of the available space, and therefore lead to implementations that might fail to interoperate. The following paragraphs set up a set of rules for using the available extension space that are designed to ensure that no such incompatibilities arise.

In the rest of this document, we call "base protocol" the protocol defined in RFC 6126, and "extended protocol" any extension of the Babel protocol that follows the rules set out in this document.

2. Mechanisms for extending the Babel protocol

2.1. New versions of the Babel protocol

The header of a Babel packet contains an eight-bit protocol version. The currently deployed version of Babel is version 2; any packets containing a version number different from 2 MUST be silently ignored.

Versions 0 and 1 were experimental versions of the Babel protocol that have seen some modest deployment; these version numbers SHOULD NOT be reused by future versions of the Babel protocol. Version numbers larger than 2 might be used by future versions should it be found necessary to define a non-backwards compatible version of the protocol.

2.2. New TLVs

An extension may carry its data in a new TLV type. Such new TLVs will be silently ignored by implementations of the base Babel protocol, as well as by other extended implementations of the Babel protocol, as long as the TLV types do not collide.

All new TLVs MUST have the format defined in RFC 6126, Section 4.3. Additionally, they SHOULD be self-terminating, in the sense defined in the next section, and any data found after the main data section of the TLV SHOULD be treated as a series of sub-TLVs.

An extension may be assigned one or more extension TLV types. A registry of known TLV types is being maintained by Juliusz Chroboczek.

2.3. Sub-TLVs

All Babel TLVs carry an explicit length. In addition, all Babel TLVs, whether defined by the base protocol or by extensions, are self-terminating, in the sense that their actual length (the "base length") can be determined without reference to the explicit length. In some cases, the base length is trivial to determine: for example, a HELLO TLV always has a base length of 2 (4 including the Type and Length fields). In other cases, determining the base length is not that easy, but is done in any case by any implementation that knows the given TLV: for example, the base length of an Update TLV depends on both the prefix length and the amount of compression being performed.

If the explicit length of a TLV is larger than its base length, the extra space present in the TLV is silently ignored by an

implementation of the base protocol, and is used by extended implementations to store a sequence of sub-TLVs. Unlike TLVs, the sub-TLVs themselves need not be self-terminating.

An extension may be assigned one or more sub-TLV types. Sub-TLV types are assigned independently from TLV types: the same numeric type can be assigned to a TLV and a sub-TLV used by different extensions. Sub-TLV types are assigned globally: once an extension is assigned a given sub-TLV number, it may use this number within any TLV; however, the meaning of a given sub-TLV type may depend on the TLV it is embedded within.

A registry of known sub-TLV types is being maintained by Juliusz Chroboczek.

2.3.1. Format of sub-TLVs

Except for Pad1 (see below), a Sub-TLV has exactly the same structure as a Babel TLV:

```
 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 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |     Body...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Fields :

Type   The type of the sub-TLV.

Length   The length of the body, exclusive of the Type and Length fields.

Body   The TLV body, the interpretation of which depends on the type.

2.3.2. Standard sub-TLVs

This document defines two types of sub-TLVs, Pad1 and PadN. These two sub-TLVs MUST be correctly parsed and ignored by any extended implementation of the Babel protocol that uses sub-TLVs.

2.3.2.1. Pad1

```
 0
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|   Type = 0    |
+-+-+-+-+-+-+-+-+
```

Fields :

Type   Set to 0 to indicate a Pad1 TLV.

This sub-TLV is silently ignored on reception.

2.3.2.2. PadN

```
 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 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type = 1   |     Length    |      MBZ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Fields :

Type   Set to 1 to indicate a PadN TLV.

Length   The length of the body, exclusive of the Type and Length fields.

MBZ   Set to 0 on transmission.

This sub-TLV is silently ignored on reception.

2.3.3. Unknown sub-TLVs

Any unknown sub-TLV MUST be silently ignored by an extended implementation that uses sub-TLVs.

2.4. The Flags field

The Flags field is an eight-bit field in the Update TLV. Bits with values 80 and 40 hexadecimal are defined in the base protocol, and MUST be recognised and used by every implementation. The remaining six bits are not currently used, and are silently ignored by existing implementations.

Extensions to the Babel protocol MAY use the six unused bits of the Flags field. However, due to the small size of the Flags field, they

SHOULD use a sub-TLV in preference to a new flag. No registry of flag assignments is currently defined.

2.5. Packet trailer

A Babel packet carries an explicit length in its header. A Babel packet is carried by a UDP datagram, which in turn contains an explicit length in its header.

It is possible for a UDP datagram carrying a Babel packet to be larger than needed to contain the packet. In that case, the extra space after the Babel packet is known as the packet trailer, and is silently ignored by an implementation of the base protocol.

The packet trailer was originally intended to be used as a cryptographic trailer. However, the authentication extension to Babel [AUTH] ended up using a pair of new TLVs, and no currently deployed extension of Babel uses the packet trailer.

3. Choosing between extension mechanisms

New versions of the Babel protocol should only be defined if the new version is not backwards compatible with the current base protocol.

In many cases, an extension could be implemented either by defining a new TLV, or by adding a new sub-TLV to an existing TLV. In particular, the most common purpose of extensions is to attach additional data to routing updates; such extensions are either implemented by creating a new "enriched" Update TLV, or by adding a sub-TLV to the Update TLV.

The two encodings are treated differently by implementations that do not understand the extension. In the case of a new TLV, the whole unknown TLV is ignored by a base implementation, while in the case of a new sub-TLV, the TLV is parsed and acted upon, and the unknown sub-TLV is silently ignored. Therefore, a sub-TLV should be preferred by extensions that extend the Update in a compatible manner (the extension data may be silently ignored), while a new TLV must be used by extensions that make incompatible extensions to the meaning of the TLV (the whole piece of data must be thrown away if the extension data is not understood).

Using a new bit in the Flags field is equivalent to defining a new sub-TLV, but uses less space in the Babel packet. Due to the high risk of collision in the limited Flags space, and the doubtful space savings, we do not recommend the use of the Flags field in future extensions.

This document refrains from making any recommendations about the usage of the packet trailer due to the lack of implementation experience.

4. References

[AUTH] Ovsienko, D., "Babel HMAC Cryptographic Authentication", Internet Draft draft-ovsienko-babel-hmac-authentication-03, April 2013.

[BABEL] Chroboczek, J., "The Babel Routing Protocol", RFC 6126, February 2011.

Author's Address

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

Email: jch@pps.univ-paris-diderot.fr