Network Working Group
Internet-Draft
Updates: 6126 (if approved)
Intended status: Experimental
Expires: January 1, 2015

J. Chroboczek
PPS, University of Paris-Diderot

June 30, 2014

Extension Mechanism for the Babel Routing Protocol
draft-chroboczek-babel-extension-mechanism-01

Abstract

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

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 http://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 1, 2015.

Table of Contents

1. Extending the Babel routing protocol

A Babel packet [BABEL] contains a header followed by a sequence of TLVs, each of which is a sequence of octets having an explicit type and length. The original 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);
- an unknown TLV MUST be silently ignored ([BABEL], Section 4.3);
- except for Pad1 and PadN, 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 conventions, independently developed extensions to the Babel protocol might make conflicting uses of the available space, and therefore lead to implementations that would fail to interoperate. This memo formalises the set of rules for extending the Babel protocol that are designed to ensure that no such incompatibilities arise, and that are currently respected by a number of deployed extensions.

In the rest of this document, we call "original 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 a future incompatible 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 original 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. New TLVs 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.

2.3. Sub-TLVs

With the exception of the Pad1 TLV, all Babel TLVs carry an explicit length. With the exception of Pad1 and PadN, all TLVs defined by the original protocol are self-terminating, in the sense that the length of the meaningful data that they contain (the "natural length") can be determined without reference to the explicitly encoded length. In some cases, the natural length is trivial to determine: for example, a HELLO TLV always has a natural length of 2 (4 including the Type and Length fields). In other cases, determining the natural length is not that easy, but needs to be done in any case by an implementation that interprets the given TLV: for example, the natural length of an Update TLV depends on both the prefix length and the amount of prefix compression being performed.

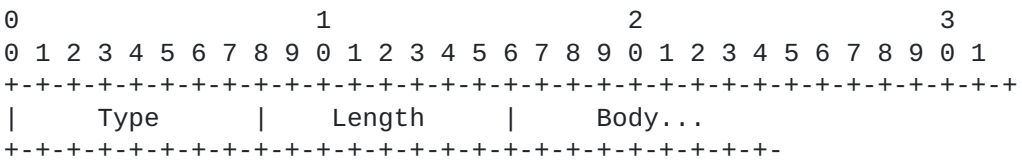
If the explicit length of a TLV is larger than its natural length, the extra space present in the TLV is silently ignored by an implementation of the original protocol; extended implementations MAY

use it to store arbitrary data, and SHOULD structure the additional data as 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 interpretation of a given sub-TLV type may depend on which particular TLV it is embedded within.

2.3.1. Format of sub-TLVs

A sub-TLV has exactly the same structure as a TLV. Except for Pad1 (see below), all sub-TLVs have the following structure:



Fields :

Type The type of the sub-TLV.

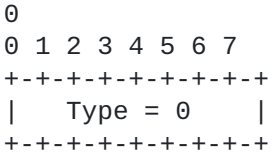
Length The length of the body, exclusive of the Type and Length fields.

Body The sub-TLV body, the interpretation of which depends on both the type of the sub-TLV and the type of the TLV within which it is embedded.

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

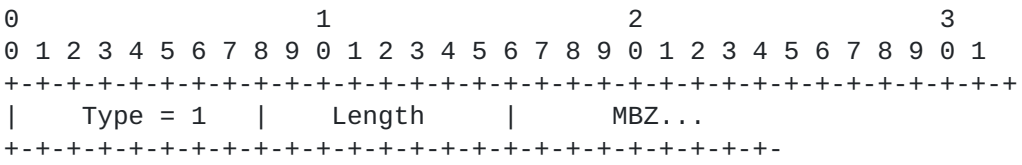


Fields :

Type Set to 0 to indicate a Pad1 sub-TLV.

This sub-TLV is silently ignored on reception.

2.3.2.2. PadN



Fields :

Type Set to 1 to indicate a PadN sub-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 by the original 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 being 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 the size of the Babel packet. In that case, the extra space after the Babel packet, known as the packet trailer, is silently ignored by an implementation of the original 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. The format and purpose of the packet trailer is therefore currently left undefined.

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 original 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. For example, an extension whose purpose is to attach additional data to route updates can be implemented either 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 an implementation of the original protocol, 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 used 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 TLV 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 while using 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. IANA Considerations

IANA is to create two new registries, called "Babel TLV types" and "Babel sub-TLV types".

The initial value of the Babel TLV types registry is as follows:

Type	Name	Reference
0	Pad1	[BABEL]
1	PadN	[BABEL]
2	Acknowledgment Request	[BABEL]
3	Acknowledgment	[BABEL]
4	Hello	[BABEL]
5	IHU	[BABEL]
6	Router-Id	[BABEL]
7	Next Hop	[BABEL]
8	Update	[BABEL]
9	Route Request	[BABEL]
10	Seqno Request	[BABEL]
11	TS/PC	[AUTH]
12	HMAC	[AUTH]
13	Source-specific Update	(Boutier)
14	Source-specific Request	(Boutier)
15	Source-specific Seqno Request	(Boutier)

The initial value of the Babel sub-TLV types registry is as follows:

Type	Name	Reference
0	Pad1	(this document)
1	PadN	(this document)
2	Diversity	(Chroboczek)
3	Timestamp	(Jonglez)

5. References

[AUTH] Ovsienko, D., "Babel HMAC Cryptographic Authentication", Internet Draft draft-ovsienko-babel-hmac-authentication-09, April 2014.

[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