Delay-Tolerant Networking Metadata Extension Block
draft-irtf-dtnrg-bundle-metadata-block-05

Abstract

This document defines an extension block that may be used with the DTN Bundle Protocol. This Metadata Extension Block is designed to carry additional information that DTN nodes can use to make processing decisions regarding bundles, such as deciding whether to store a bundle or determining to which nodes to forward a bundle. The metadata that is carried in a metadata block must be formatted according to the metadata type that is identified in the block’s metadata type field. One specific metadata type, for carrying URIs as metadata, is defined in this document. Other metadata types may be defined in separate documents.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on May 10, 2010.

Copyright Notice

Copyright (c) 2009 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 (http://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 BSD License.

Table of Contents

1. Introduction .................................................. 3
2. Metadata Block Format ........................................ 5
3. Metadata Block Processing ..................................... 7
   3.1. Bundle Transmission ....................................... 7
   3.2. Bundle Forwarding ......................................... 7
   3.3. Bundle Reception .......................................... 7
4. Predefined Metadata Types .................................... 8
   4.1. URI Metadata Type ......................................... 8
   4.2. Private Metadata Types ................................... 8
5. Security Considerations ....................................... 9
6. IANA Considerations .......................................... 10
7. References .................................................... 11
   7.1. Normative References .................................... 11
   7.2. Informative References .................................. 11
Author’s Address .................................................. 12
1. Introduction

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 RFC 2119 [refs.RFC2119].

This document defines an extension block that may be used with the Bundle Protocol [refs.DTNBP] within the context of a Delay-Tolerant Network architecture [refs.DTNarch]. The DTN bundle protocol [refs.DTNBP] defines the bundle as its protocol data unit. This document defines a bundle block called a Metadata Block. This block is designed to carry additional information that DTN nodes can use to make processing decisions regarding bundles.

The metadata block has been deliberately defined to be flexible enough that it would be capable of supporting a variety of metadata types and formats. However, as mentioned, it is the intention that the metadata that is carried in this block be additional application-related information. For example, the metadata might be information that is associated with the payload of a bundle. Additional extension blocks could be (and have been) defined for carrying additional network-related information.

While the bundle payload may be processed opaquely by DTN nodes, metadata is intended to serve as a mechanism for providing DTN nodes with access to additional information that they can use to process the bundle. Examples of such additional information include keywords found in the payload, payload provenance information, GPS coordinates (if the payload is a map, for instance), message IDs, and artist, album and track name (if the payload is a song). Even though the metadata is additional information related to the application, its purpose is to be used by DTN nodes to make decisions regarding how to process bundles within the network, such as whether or not a bundle should be stored or to which nodes a bundle should be forwarded. Metadata that is about bundle payload, for example, might serve as a content-based index of bundles that are stored in a DTN cache. So, in response to a request for bundles related to a certain subject or related to specific GPS coordinates, for example, the metadata of stored bundles could be searched and all bundles with metadata matching the search criteria could be retrieved and returned to the requestor.

This document defines the general format of and the processing required to support the Metadata Block. The actual metadata to be inserted into a metadata block MUST be formatted according to the metadata type that is identified in the block's metadata type field. One specific metadata type, for carrying Uniform Resource Identifiers
(URIs) [refs.URI] as metadata, is defined in this document. Other metadata types may be defined in separate documents, along with the steps required to process records of that type, if necessary. If such other metadata types are defined, they should be registered to ensure global uniqueness (see the IANA Considerations section).

A bundle may contain multiple metadata extension blocks. In some cases, multiple metadata blocks may be carried in the bundle, possibly with each being encrypted separately from each other and from the payload. Such separate encryption of metadata from payload would enable bundle nodes to perform content-based searching and routing on bundle metadata that they are able to decrypt, even if they are not able to decrypt the bundle payload.

The capabilities described in this document are OPTIONAL for deployment with the Bundle Protocol. Bundle Protocol implementations claiming to support the Metadata Block MUST be capable of:

-Generating a Metadata Block and inserting it into a bundle,

-Receiving bundles containing a Metadata Block and making the information contained in this Metadata Block’s metadata field available for use, e.g., in bundle storage or forwarding decisions,

-Deleting a metadata block from a received bundle before forwarding the bundle

as defined in this document.

Bundle Protocol implementations claiming to support a specific metadata type must both support the metadata block as defined above and be capable of parsing and processing the metadata itself according to the definition of the metadata type in which the metadata is expressed. This metadata type may be the URI Metadata Type (see the URI Metadata Type section), or it may be another metadata type defined in a separate document.
2. Metadata Block Format

The Metadata Block uses the Canonical Bundle Block Format as defined in the bundle protocol [refs.DTNBP]. That is, it is comprised of the following elements:

- Block-type code (1 byte) - defined as in all bundle protocol blocks except the primary bundle block (as described in the Bundle Protocol). The block type code for the Metadata Block is 0x08.

- Block processing control flags (SDNV) - defined as in all bundle protocol blocks except the primary bundle block. SDNV encoding is described in the Bundle Protocol. There are no constraints on the use of the Block Processing Control Flags. If a bundle node receives a bundle with a metadata block and it is capable of supporting the metadata block but it is not able to parse and process the metadata itself, either because it does not support the metadata type being used or because the metadata is not well-formed according to the metadata type definition, the bundle node must process the bundle as if it cannot process the metadata block. That is, it must operate according to the settings of the Block Processing Control Flags, including the "Delete bundle if block can’t be processed" flag and the "Discard block if it can’t be processed" flag.

- EID Reference Count and List (optional) - composite field defined in the bundle protocol that is present if and only if the metadata block references EID elements in the primary block’s dictionary. Presence of this field is indicated by the setting of the "Block contains an EID-reference field" bit of the block processing control flags. If EIDs are referenced in the metadata block, then their interpretation is defined by the particular metadata type that is being used in this metadata block, as indicated in the metadata type field.

- Block data length (SDNV) - defined as in all bundle protocol blocks except the primary bundle block. SDNV encoding is described in the bundle protocol.

- Block-type-specific data fields as follows:

  - Metadata Type field (SDNV) - indicates which metadata type is to be used to interpret both the metadata in the metadata field and the EID references in the optional EID Reference Count and List field (if present). One metadata type is defined in this document. Other metadata types may be defined in separate documents.
- Metadata field - contains the metadata itself, formatted according to the metadata type that has been specified for this block.

The Structure of a Metadata Block is as follows:

```
Metadata Block Format:
+-----+------+--------------------+------+----------+----------|
|Type | Flags |EID Reference count |Len   | Metadata | Metadata |
   |     | (SDNV) | and list (opt)    |(SDNV)|   Type   |          |
+-----+------+--------------------+------+----------+----------+
```

Figure 1
3. Metadata Block Processing

The following are the processing steps that a bundle node may take relative to generation, reception, and processing of Metadata Blocks.

3.1. Bundle Transmission

When an outbound bundle is created per the parameters of the bundle transmission request, this bundle MAY (as influenced by local policy and the metadata type being used) include one or more metadata blocks (as defined in this specification).

3.2. Bundle Forwarding

A node MAY insert one or more metadata Blocks into a bundle before forwarding it; and a node MAY delete one or more metadata blocks from a bundle before forwarding it, as dictated by local policy and the metadata type being used.

3.3. Bundle Reception

If the bundle includes one or more metadata blocks, the metadata information records in these blocks SHALL be made available for use at this node (e.g., in bundle storage or forwarding decisions, or, if the receiving node is the bundle destination, the metadata information records may be provided to the receiving application).
4. Predefined Metadata Types

As mentioned in the previous section, any number of different metadata types may be defined to indicate the format of both the metadata field and the EID references in the optional EID Reference Count and List field (if present) and, if necessary, how metadata of this type should be processed. One metadata type is defined in this document, URI Metadata Type (0x01). In addition, a range of metadata type values is reserved for private use.

4.1. URI Metadata Type

It is believed that use of URIs will in many cases be adequate for encoding metadata, although it is recognized that use of URIs may not be the most efficient method for such encoding. Because of the expected utility of using URI encoding for metadata, the metadata type value of 0x01 is defined to indicate a metadata type of URI. Metadata type values other than 0x01 will be used to indicate alternative metadata types.

The Metadata field for metadata of metadata type URI (0x01) consists of an array of bytes formed by concatenating one or more null-terminated URIs. Unless determined by local policy, the specific processing steps that must be performed on bundles with metadata blocks containing metadata of type URI are expected to be included as part of the URI encoding of the metadata. Metadata blocks containing metadata of type URI MUST NOT include an EID references field. The absence of this field MUST be indicated by a value of 0 for the "block contains an EID reference field" flag in the block processing control flags. Support for the URI metadata type is OPTIONAL.

4.2. Private Metadata Types

Metadata type values 192 through 249 are not defined in this specification and are available for private and/or experimental use. Such private metadata types and are not required to be registered. All other values of the metadata type are reserved for future use and, when defined, should be registered to ensure global uniqueness (see the IANA Considerations section). Local policy will define how private metadata types are handled.
5. Security Considerations

The DTN Bundle Security Protocol [refs.DTNBPsec] defines security-related blocks to provide hop-by-hop authentication, end-to-end authentication, end-to-end confidentiality of bundles or parts of bundles, and an extension security block to provide confidentiality and integrity for extension blocks, as well as a set of standard ciphersuites that may be used to calculate security results carried in these security blocks. All ciphersuites that use the strict canonicalisation algorithm [refs.DTNBPsec] to calculate and verify security results (e.g., many hop-by-hop authentication ciphersuites) apply to all blocks in the bundle, and so would apply to bundles that include an optional Metadata Block and would include that block in the calculation of their security result. In particular, bundles including the optional Metadata Block would be protected in their entirety for the duration of a single hop, from a forwarding node to an adjacent receiving node (but not from source to destination), using the standard BAH-HMAC ciphersuite defined in the Bundle Security Protocol. Ciphersuites that use the mutable canonicalisation algorithm to calculate and verify security results (e.g., the mandatory PSH-RSA-SHA256 ciphersuite and most end-to-end authentication ciphersuites) will omit the Metadata Block from their calculation. The fact that metadata in the metadata block may be modified or that metadata blocks themselves may be added to or deleted from a bundle as it transits the network will not interfere with end-to-end security protection when using ciphersuites that use mutable canonicalisation. Lastly, the Metadata Block will not be encrypted by the mandatory CH-RSA-AES-PAYLOAD-PSH end-to-end confidentiality ciphersuite, which only allows for payload and PSH encryption.

In order to provide the metadata block with end-to-end confidentiality and authentication independent of any confidentiality or authentication that is provided for the payload or other parts of the bundle, the extension security block may be used to encrypt and authenticate the metadata block.

Given that metadata can be modified by forwarding nodes, it may be desirable to eventually support the ability to audit changes to the metadata at the individual record level. No such capability has been provided in this specification as currently written.
6. IANA Considerations

We may want to consider having IANA establish a register of Bundle Protocol header types, with the Metadata Extension Block header identified as type 0x08. In association with the Metadata Extension block, we may want IANA to establish a separate register of metadata types. The metadata type code used for the URI metadata type is 0x01. Metadata type values 192 through 249 are used for private and/or experimental use.
7. References

7.1. Normative References

[refs.RFC2119]

[refs.URI]

[refs.DTNBP]

[refs.DTNBPsec]

7.2. Informative References

[refs.DTNarch]
Author’s Address

Susan Flynn Symington
The MITRE Corporation
7515 Colshire Drive
McLean, VA  22102
US

Phone: +1 (703) 983-7209
Email: susan@mitre.org
URI:  http://mitre.org/