IRTF Delay Tolerant Networking
Research Group
Internet-Draft
Intended status: Experimental
Expires: January 4, 2008

K. Scott
The MITRE Corporation
S. Burleigh
NASA Jet Propulsion Laboratory
July 3, 2007

Bundle Protocol Specification
draft-irtf-dtnrg-bundle-spec-10.txt

Status of this Memo

Copyright Notice

Abstract

This document describes the end-to-end protocol, block formats, and abstract service description for the exchange of messages (bundles) in Delay Tolerant Networking (DTN).

This document was produced within the IRTF's Delay Tolerant Networking Research Group (DTNRG) and represents the consensus of all of the active contributors to this Group. See http://www.dtnrg.org for more information.

Table of Contents

1. Requirements notation

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].

2. Introduction

This document describes version 6 of the Delay Tolerant Networking (DTN) "bundle" protocol (BP). Delay Tolerant Networking is an end-to-end architecture providing communications in and/or through highly stressed environments. Stressed networking environments include those with intermittent connectivity, large and/or variable delays, and high bit error rates. To provide its services, BP sits at the application layer of some number of constituent internets, forming a store-and-forward overlay network. Key capabilities of BP include:

- Custody-based retransmission
- Ability to cope with intermittent connectivity
- Ability to take advantage of scheduled, predicted, and opportunistic connectivity (in addition to continuous connectivity)
- Late binding of overlay network endpoint identifiers to constituent internet addresses

For descriptions of these capabilities and the rationale for the DTN architecture, see [ARCH] and [SIGC]. [TUT] contains a tutorial-level overview of DTN concepts.

This is an experimental protocol, produced within the IRTF's Delay Tolerant Networking Research Group (DTNRG) and represents the consensus of all of the active contributors to this Group. If this protocol is used on the Internet, IETF standard protocols for security and congestion control should be used.

BP's location within the standard protocol stack is as shown in Figure 1. BP uses the "native" internet protocols for communications within a given internet. Note that "internet" in the preceding is used in a general sense and does not necessarily refer to TCP/IP. The interface between the common bundle protocol and a specific internetwork protocol suite is termed a "convergence layer adapter". Figure 1 shows three distinct transport and network protocols (denoted T1/N1, T2,N2, and T3/N3).

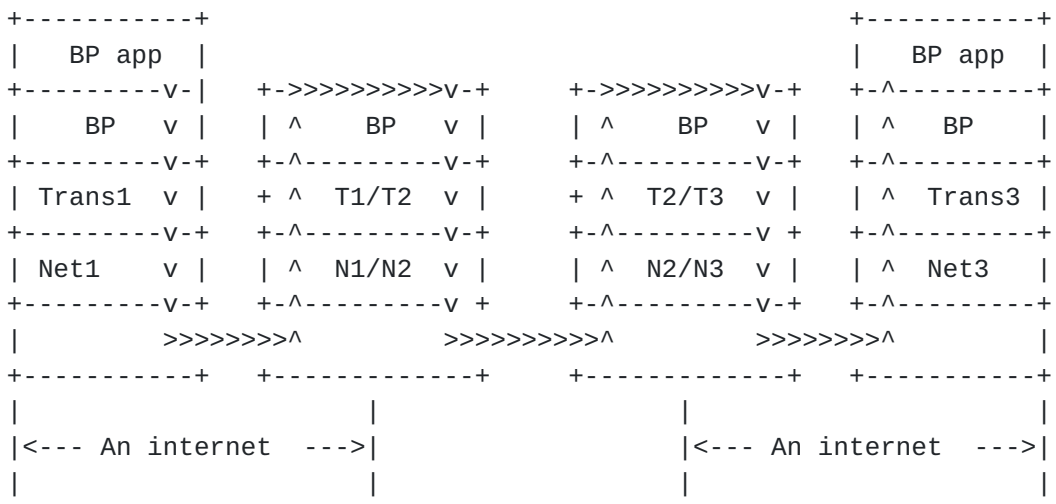


Figure 1: The bundle protocol sits at the application layer of the Internet model.

This document describes the format of the protocol data units (called bundles) passed between entities participating in BP communications. The entities are referred to as "bundle nodes". This document does not address:

- Operations in the convergence layer adapters that bundle nodes use to transport data through specific types of internet. (However, the document does discuss the services that must be provided by each adapter at the convergence layer.)

- The bundle routing algorithm.

- Mechanisms for populating the routing or forwarding information bases of bundle nodes.

3. Service Description

3.1. Definitions

Bundle - A bundle is a protocol data unit of the DTN bundle protocol. Each bundle comprises a sequence of two or more "blocks" of protocol data, which serve various purposes. Multiple instances of the same bundle (the same unit of DTN protocol data) might exist concurrently in different parts of a network - possibly in different representations - in the memory local to one or more bundle nodes and/or in transit between nodes. In the context of the operation of a bundle node, a bundle is an instance of some bundle in the network that is in that node's local memory.

Bundle payload - A bundle payload (or simply "payload") is the application data whose conveyance to the bundle's destination is the purpose for the transmission of a given bundle. The terms "bundle content", "bundle payload", and "payload" are used interchangeably in this document. The "nominal" payload for a bundle forwarded in response to a bundle transmission request is the application data unit whose location is provided as a parameter to that request. The nominal payload for a bundle forwarded in response to reception of that bundle is the payload of the received bundle.

Fragment - A fragment is a bundle whose payload block contains a fragmentary payload. A fragmentary payload is either the first N bytes or the last N bytes of some other payload - either a nominal payload or a fragmentary payload - of length M, such that 0 < N < M.

Bundle node - A bundle node (or, in the context of this document, simply a "node") is any entity that can send and/or receive bundles. In the most familiar case a bundle node is instantiated as a single process running on a general-purpose computer, but in general the definition is meant to be broader: a bundle node might alternatively be a thread, an object in an object-oriented operating system, a special-purpose hardware device, etc. Each bundle node has three conceptual components, defined below: a "bundle protocol agent", a set of zero or more "convergence layer adapters", and an "application agent".

Bundle protocol agent - The bundle protocol agent (BPA) of a node is the node component that offers the BP services and executes the procedures of the Bundle Protocol. The manner in which it does so is wholly an implementation matter. For example, BPA functionality might be coded into each node individually; it might be implemented as a shared library that is used in common by any

number of bundle nodes on a single computer; it might be implemented as a daemon whose services are invoked via inter-process or network communication by any number of bundle nodes on one or more computers; it might be implemented in hardware.

Convergence layer adapters - A convergence layer adapter (CLA) sends and receives bundles on behalf of the BPA, utilizing the services of some 'native' internet protocol that is supported in one of the internets within which the node is functionally located. The manner in which a CLA sends and receives bundles is wholly an implementation matter, exactly as described for the BPA.

Application agent - The application agent (AA) of a node is the node component that utilizes the BP services to effect communication for some purpose. The application agent in turn has two elements, an administrative element and an application-specific element. The application-specific element of an AA constructs, requests transmission of, accepts delivery of, and processes application-specific application data units; the only interface between the BPA and the application-specific element of the AA is the BP service interface. The administrative element of an AA constructs and requests transmission of administrative records (status reports and custody signals), and it accepts delivery of and processes any custody signals that the node receives. In addition to the BP service interface, there is a (conceptual) private control interface between the BPA and the administrative element of the AA that enables each to direct the other to take action under specific circumstances. In the case of a node that serves simply as a "router" in the overlay network, the AA may have no application-specific element at all. The application-specific elements of other nodes' AAs may perform arbitrarily complex application functions, perhaps even offering multiplexed DTN communication services to a number of other applications. As with the BPA, the manner in which the AA performs its functions is wholly an implementation matter; in particular, the administrative element of an AA might be built into the library or daemon or hardware that implements the BPA, and the application-specific element of an AA might be implemented either in software or in hardware.

Bundle endpoint - A bundle endpoint (or simply "endpoint") is a set of zero or more bundle nodes that all identify themselves for BP purposes by some single text string, called a "bundle endpoint ID" (or, in this document, simply "endpoint ID"; endpoint IDs are described in detail in 3.4 below). The special case of an endpoint that never contains more than one node is termed a "singleton" endpoint; every bundle node must be a member of at least one singleton endpoint. Singletons are the most familiar

sort of endpoint, but in general the endpoint notion is meant to be broader. For example, the nodes in a sensor network might constitute a set of bundle nodes that identify themselves by a single common endpoint ID and thus form a single bundle endpoint. **Note** too that a given bundle node might identify itself by multiple endpoint IDs and thus be a member of multiple bundle endpoints.

Forwarding - When the bundle protocol agent of a node determines that a bundle must be "forwarded" to an endpoint, it causes the bundle to be sent to all of the nodes that the bundle protocol agent currently believes are in the "minimum reception group" of that endpoint. The minimum reception group of an endpoint may be any one of the following: (a) ALL of the nodes registered in an endpoint that is permitted to contain multiple nodes (in which case forwarding to the endpoint is functionally similar to "multicast" operations in the Internet, though possibly very different in implementation); (b) ANY N of the nodes registered in an endpoint that is permitted to contain multiple nodes, where N is in the range from zero to the cardinality of the endpoint (in which case forwarding to the endpoint is functionally similar to "anycast" operations in the Internet); (c) THE SOLE NODE registered in a singleton endpoint (in which case forwarding to the endpoint is functionally similar to "unicast" operations in the Internet). The nature of the minimum reception group for a given endpoint can be determined from the endpoint's ID (again, see 3.4 below): for some endpoint ID "schemes", the nature of the minimum reception group is fixed - in a manner that is defined by the scheme - for all endpoints identified under the scheme; for other schemes, the nature of the minimum reception group is indicated by some lexical feature of the "scheme-specific part" of the endpoint ID, in a manner that is defined by the scheme.

Registration - A registration is the state machine characterizing a given node's membership in a given endpoint. Any number of registrations may be concurrently associated with a given endpoint, and any number of registrations may be concurrently associated with a given node. Any single registration must at any time be in one of two states: Active, Passive. A registration always has an associated "delivery failure action", the action that is to be taken when a bundle that is "deliverable" (see below) subject to that registration is received at a time when the registration is in the Passive state. Delivery failure action must be one of the following:

* defer "delivery" (see below) of the bundle subject to this registration until (a) this bundle is the least recently received of all bundles currently deliverable subject to this

registration and (b) either the registration is polled or else the registration is in Active state;

* "abandon" (see below) delivery of the bundle subject to this registration.

An additional implementation-specific delivery deferral procedure may optionally be associated with the registration. While the state of a registration is Active, reception of a bundle that is deliverable subject to this registration must cause the bundle to be delivered automatically as soon as it is the least recently received bundle that is currently deliverable subject to the registration. While the state of a registration is Passive, reception of a bundle that is deliverable subject to this registration must cause delivery of the bundle to be abandoned or deferred as mandated by the registration's current delivery failure action; in the latter case, any additional delivery deferral procedure associated with the registration must also be performed.

Delivery - Upon reception, the processing of a bundle that has been sent to a given node depends on whether or not the receiving node is registered in the bundle's destination endpoint; if it is, and if the payload of the bundle is non-fragmentary (possibly as a result of successful payload reassembly from fragmentary payloads, including the original payload of the received bundle), then the bundle is normally "delivered" to the node's application agent subject to the registration characterizing the node's membership in the destination endpoint. A bundle is considered to have been delivered at a node subject to a registration as soon as the application data unit that is the payload of the bundle, together with the value of the bundle's "Acknowledgement by application is requested" flag and any other relevant metadata (an implementation matter), has been presented to the node's application agent in a manner consistent with the state of that registration and, as applicable, the registration's delivery failure action.

Deliverability, Abandonment - A bundle is considered "deliverable" subject to a registration if and only if (a) the bundle's destination endpoint is the endpoint with which the registration is associated, (b) the bundle has not yet been delivered subject to this registration, and (c) delivery of the bundle subject to this registration has not been abandoned. To "abandon" delivery of a bundle subject to a registration is simply to declare it no longer deliverable subject to that registration; normally only registrations' registered delivery failure actions cause deliveries to be abandoned.

Deletion, Discarding - A bundle protocol agent "discards" a bundle by simply ceasing all operations on the bundle and functionally erasing all references to it; the specific procedures by which this is accomplished are an implementation matter. Bundles are discarded silently, i.e., the discarding of a bundle does not result in generation of an administrative record. "Retention constraints" are elements of bundle state that prevent a bundle from being discarded; a bundle cannot be discarded while it has any retention constraints. A bundle protocol agent "deletes" a bundle in response to some anomalous condition by notifying the bundle's report-to endpoint of the deletion (provided such notification is warranted; see 4.13 for details) and then arbitrarily removing all of the bundle's retention constraints, enabling the bundle to be discarded.

Transmission - A transmission is a sustained effort by a node's bundle protocol agent to cause a bundle to be sent to all nodes in the minimum reception group of some endpoint (which may be the bundle's destination or may be some intermediate forwarding endpoint) in response to a transmission request issued by the node's application agent. Any number of transmissions may be concurrently undertaken by the bundle protocol agent of a given node.

Custody - To "accept custody" upon forwarding a bundle is to commit to retaining a copy of the bundle - possibly re-forwarding the bundle when the necessity to do so is determined - until custody of that bundle is "released". Custody of a bundle whose destination is a singleton endpoint is released when either (a) notification is received that some other node has accepted custody of the same bundle, (b) notification is received that the bundle has been delivered at the (sole) node registered in the bundle's destination endpoint, or (c) the bundle is explicitly deleted for some reason, such as lifetime expiration. The condition(s) under which custody of a bundle whose destination is not a singleton endpoint may be released are not defined in this specification. To "refuse custody" of a bundle is to decide not to accept custody of the bundle. A "custodial node" of a bundle is a node that has accepted custody of the bundle and has not yet released that custody. A "custodian" of a bundle is a singleton endpoint whose sole member is one of the bundle's custodial nodes.

3.2. Implementation architectures

The above definitions are intended to enable the bundle protocol's operations to be specified in a manner that minimizes bias toward any particular implementation architecture. To illustrate the range of interoperable implementation models that might conform to this

specification, four example architectures are briefly described below.

1. Bundle protocol application server

   A single bundle protocol application server, constituting a single bundle node, runs as a daemon process on each computer. The daemon's functionality includes all functions of the bundle protocol agent, all convergence layer adapters, and both the administrative and application-specific elements of the application agent. The application-specific element of the application agent functions as a server, offering bundle protocol service over a local area network: it responds to remote procedure calls from application processes (on the same computer and/or remote computers) that need to communicate via the bundle protocol. The server supports its clients by creating a new (conceptual) node for each one and registering each such node in a client-specified endpoint; the conceptual nodes managed by the server function as clients' Bundle Protocol service access points.

2. Peer application nodes

   Any number of bundle protocol application processes, each one constituting a single bundle node, run in ad-hoc fashion on each computer. The functionality of the bundle protocol agent, all convergence layer adapters, and the administrative element of the application agent is provided by a library to which each node process is dynamically linked at run time; the application-specific element of each node's application agent is node-specific application code.

3. Sensor network nodes

   Each node of the sensor network is the self-contained implementation of a single bundle node. All functions of the bundle protocol agent, all convergence layer adapters, and the administrative element of the application agent are implemented in simplified form in ASICs, while the application-specific element of each node's application agent is implemented in a programmable microcontroller. Forwarding is rudimentary: all bundles are forwarded on a hard-coded default route.

4. Dedicated bundle router

   Each computer constitutes a single bundle node that functions solely as a high-performance bundle forwarder. Many standard functions of the bundle protocol agent, the convergence layer

adapters, and the administrative element of the application agent are implemented in ASICs, but some functions are implemented in a high-speed processor to enable reprogramming as necessary. The node's application agent has no application-specific element. Substantial non-volatile storage resources are provided, and arbitrarily complex forwarding algorithms are supported.

3.3. Services offered by bundle protocol agents

The bundle protocol agent of each node is expected to provide the following services to the node's application agent:

- commencing a registration (registering a node in an endpoint);
- terminating a registration;
- switching a registration between Active and Passive state;
- transmitting a bundle to an identified bundle endpoint;
- canceling a transmission;
- polling a registration that is in passive state;
- delivering a received bundle.

4. Bundle Format

Each bundle shall be a concatenated sequence of at least two block structures. The first block in the sequence must be a primary bundle block, and no bundle may have more than one primary bundle block. Additional bundle protocol blocks of other types may follow the primary block to support extensions to the Bundle Protocol, such as the Bundle Security Protocol [BSP]. At most one of the blocks in the sequence may be a payload block. The last block in the sequence must have the "last block" flag (in its block processing control flags) set to 1; for every other block in the bundle after the primary block, this flag must be set to zero.

4.1. Self-Delimiting Numeric Values (SDNVs)

The design of the bundle protocol attempts to reconcile minimal consumption of transmission bandwidth with:

- extensibility to address requirements not yet identified, and
- scalability across a wide range of network scales and payload sizes.

A key strategic element in the design is the use of self-delimiting numeric values (SDNVs). The SDNV encoding scheme is closely adapted from the Abstract Syntax Notation One Basic Encoding Rules for subidentifiers within an object identifier value [ASN1]. An SDNV is a numeric value encoded in N octets, the last of which has its most significant bit (MSB) set to zero; the MSB of every other octet in the SDNV must be set to 1. The value encoded in an SDNV is the unsigned binary number obtained by concatenating into a single bit string the 7 least significant bits of each octet of the SDNV.

The following examples illustrate the encoding scheme for various hexadecimal values.

```
0xABC  : 1010 1011 1100
         is encoded as
         {1 00 10101} {0 0111100}
         = 10010101 00111100

0x1234 : 0001 0010 0011 0100
       =    1 0010 0011 0100
         is encoded as
         {1 0 100100} {0 0110100}
         = 10100100 00110100

0x4234 : 0100 0010 0011 0100
       =  100 0010 0011 0100
         is encoded as
         {1 000000 1} {1 0000100} {0 0110100}
         = 10000001 10000100 00110100

0x7F   : 0111 1111
       =  111 1111
         is encoded as
         {0 1111111}
         = 01111111
```

Figure 2: SDNV Example

Note: Care must be taken to make sure that the value to be encoded is (in concept) padded with high-order zero bits to make its bitwise length a multiple of 7 before encoding. Also note that, while there is no theoretical limit on the size of an SDNV field, the overhead of the SDNV scheme is 1:7, i.e., one bit of overhead for every 7 bits of actual data to be encoded. Thus a 7-octet value (a 56-bit quantity with no leading zeroes) would be encoded in an 8-octet SDNV; an 8-octet value (a 64-bit quantity with no leading zeroes) would be encoded in a 10-octet SDNV (one octet containing the high-order bit of the value padded with six leading zero bits, followed by nine octets containing the remaining 63 bits of the value). 148 bits of overhead would be consumed in encoding a 1024-bit RSA encryption key directly in an SDNV. In general, an N-bit quantity with no leading zeroes is encoded in an SDNV occupying ceil(N/7) octets, where ceil is the integer ceiling function.

Implementations of the Bundle Protocol may handle as an invalid numeric value any SDNV that encodes an integer that is larger than (2^64 - 1).

An SDNV can be used to represent both very large and very small integer values. However, SDNV is clearly not the best way to represent every numeric value. For example, an SDNV is a poor way to

represent an integer whose value typically falls in the range 128 to 255. In general, though, we believe that SDNV representation of numeric values in bundle blocks yields the smallest block sizes without sacrificing scalability.

4.2. Bundle Processing Control Flags

The bundle processing control flags field in the primary bundle block of each bundle is an SDNV; the value encoded in this SDNV is a string of bits used to invoke selected bundle processing control features. The significance of the value in each currently defined position of this bit string is described here. Note that in the figure and descriptions, the bit label numbers denote position (from least significant ('0') to most significant) within the decoded bit string, and not within the representation of the bits on the wire. This is why the descriptions in this section and the next do not follow standard RFC conventions with bit 0 on the left; if fields are added in the future, the SDNV will grow to the left, and using this representation allows the references here to remain valid.

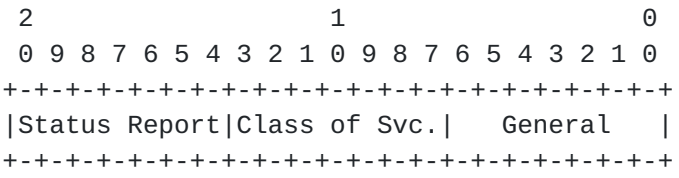


Figure 3: Bundle processing control flags bit layout

The bits in positions 0 through 6 are flags that characterize the bundle as follows:

0 -- Bundle is a fragment.

1 -- Application data unit is an administrative record.

2 -- Bundle must not be fragmented.

3 -- Custody transfer is requested.

4 -- Destination endpoint is a singleton.

5 -- Acknowledgement by application is requested.

6 -- Reserved for future use.

The bits in positions 7 through 13 are used to indicate the bundle's class of service. The bits in positions 8 and 7 constitute a two-bit priority field indicating the bundle's priority, with higher values

being of higher priority: 00 = bulk, 01 = normal, 10 = expedited, 11 is reserved for future use. Within this field, bit 8 is the most significant bit. The bits in positions 9 through 13 are reserved for future use.

The bits in positions 14 through 20 are status report request flags. These flags are used to request status reports as follows:

14 -- Request reporting of bundle reception.

15 -- Request reporting of custody acceptance.

16 -- Request reporting of bundle forwarding.

17 -- Request reporting of bundle delivery.

18 -- Request reporting of bundle deletion.

19 -- Reserved for future use.

20 -- Reserved for future use.

If the bundle processing control flags indicate that the bundle's application data unit is an administrative record, then the custody transfer requested flag must be zero and all status report request flags must be zero. If the custody transfer requested flag is 1 then the sending node requests that the receiving node accept custody of the bundle. If the bundle's source endpoint ID is "dtn:none" (see below), then the bundle is not uniquely identifiable and all bundle protocol features that rely on bundle identity must therefore be disabled: the bundle's custody transfer requested flag must be zero, the "bundle must not be fragmented" flag must be 1, and all status report request flags must be zero.

4.3. Block Processing Control Flags

The block processing control flags field in every block other than the primary bundle block is an SDNV; the value encoded in this SDNV is a string of bits used to invoke selected block processing control features. The significance of the values in all currently defined positions of this bit string, in order from least-significant position in the decoded bit string (labeled '0') to most-significant (labeled '6'), are described here.

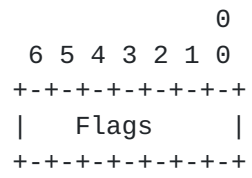


Figure 4: Block processing control flags bit layout

0 - Block must be replicated in every fragment.

1 - Transmit status report if block can't be processed.

2 - Delete bundle if block can't be processed.

3 - Last block.

4 - Discard block if it can't be processed.

5 - Block was forwarded without being processed.

6 - Block contains an EID-reference field.

For each bundle whose primary block's bundle processing control flags (see above) indicate that the bundle's application data unit is an administrative record, the "Transmit status report if block can't be processed" flag in the block processing flags field of every other block in the bundle must be zero.

The 'Block must be replicated in every fragment' bit in the block processing flags must be set to zero on all blocks that follow the payload block.

4.4. Endpoint IDs

The destinations of bundles are bundle endpoints, identified by text strings termed "endpoint IDs" (see Section 3.1). Each endpoint ID conveyed in any bundle block takes the form of a Uniform Resource Identifier (URI; [URI]). As such, each endpoint ID can be characterized as having this general structure:

< scheme name > : < scheme-specific part, or "SSP" >

As used for the purposes of the bundle protocol, neither the length of a scheme name nor the length of an SSP may exceed 1023 bytes.

Bundle blocks cite a number of endpoint IDs for various purposes of the bundle protocol. Many, though not necessarily all, of the endpoint IDs referred to in the blocks of a given bundle are conveyed

in the "dictionary" byte array in the bundle's primary block. This array is simply the concatenation of any number of null-terminated scheme names and SSPs.

"Endpoint ID references" are used to cite endpoint IDs that are contained in the dictionary; all endpoint ID citations in the primary bundle block are endpoint ID references, and other bundle blocks may contain endpoint ID references as well. Each endpoint ID reference is an ordered pair of SDNVs:

- The first SDNV contains the offset within the dictionary of the first character of the referenced endpoint ID's scheme name.

- The second SDNV contains the offset within the dictionary of the first character of the referenced endpoint ID's SSP.

This encoding enables a degree of block compression: when the source and report-to of a bundle are the same endpoint, for example, the text of that endpoint's ID may be cited twice yet appear only once in the dictionary.

The scheme identified by the < scheme name > in an endpoint ID is a set of syntactic and semantic rules that fully explain how to parse and interpret the SSP. The set of allowable schemes is effectively unlimited. Any scheme conforming to [URIREG] may be used in a bundle protocol endpoint ID. In addition, a single additional scheme is defined by the present document:

- The "dtn" scheme, which is used at minimum in the representation of the null endpoint ID "dtn:none". The forwarding of a bundle to the null endpoint is never contraindicated, and the minimum reception group for the null endpoint is the empty set.

Note that, although the endpoint IDs conveyed in bundle blocks are expressed as URIs, implementations of the BP service interface may support expression of endpoint IDs in some internationalized manner (e.g., IRIs; see RFC 3987).

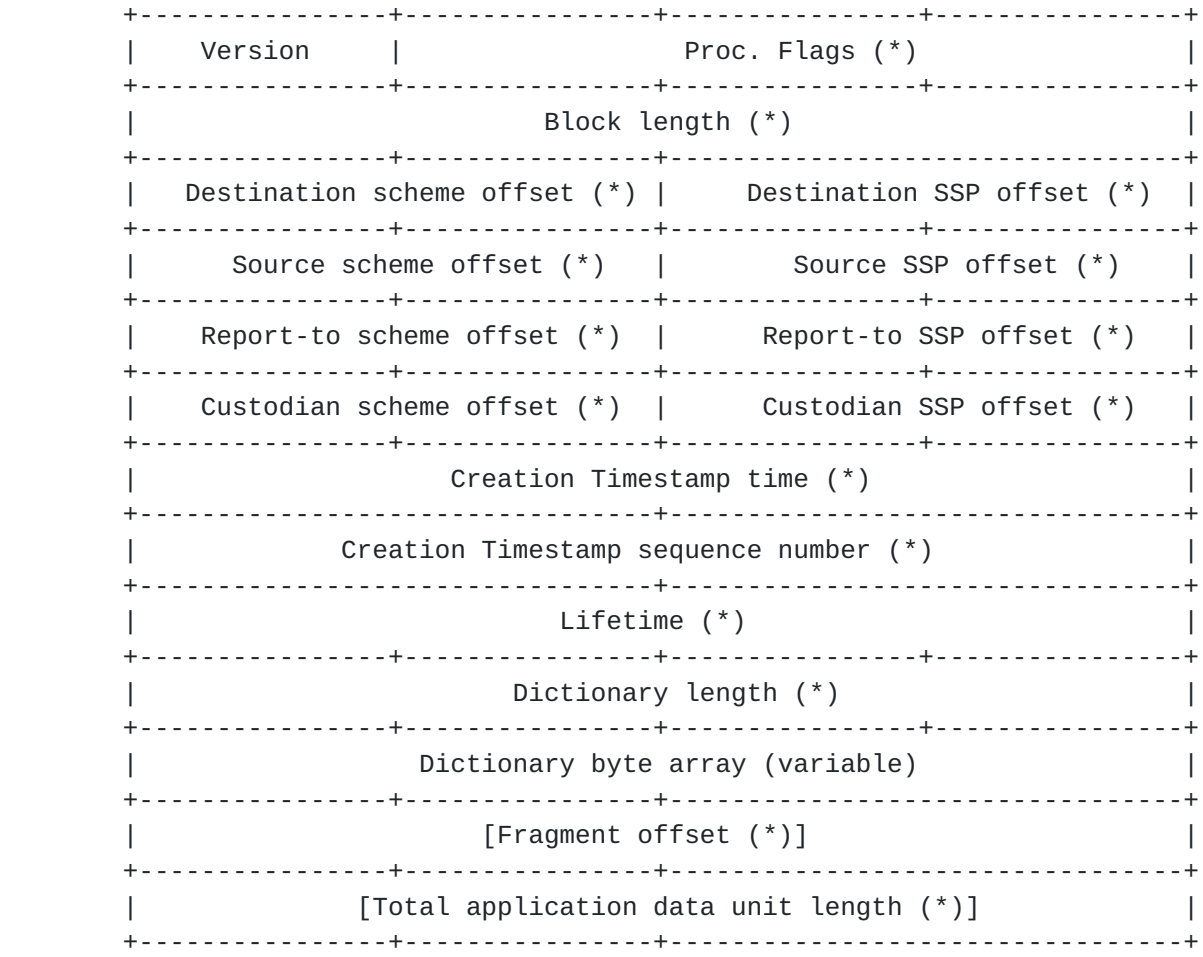
4.5. Formats of Bundle Blocks

This section describes the formats of the primary block and payload block. Rules for processing these blocks appear in Section 5 of this document.

Note that supplementary DTN protocol specifications (including, but not restricted to, the Bundle Security Protocol [BSP]) may require that BP implementations conforming to those protocols construct and process additional blocks.

The format of the two basic BP blocks is shown in Figure 5 below.

Primary Bundle Block



Bundle Payload Block

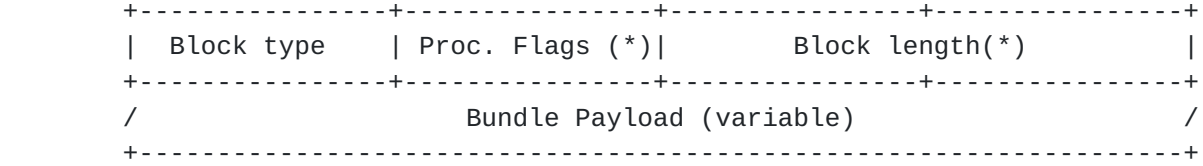


Figure 5: Bundle Block Formats

(*) Notes:

The bundle processing control ("Proc.") flags field in the Primary Bundle Block is an SDNV and is therefore variable-length. A three-octet SDNV is shown here for convenience in representation.

The block length field of the Primary Bundle Block is an SDNV and is

therefore variable-length. A four-octet SDNV is shown here for convenience in representation.

Each of the eight offset fields in the Primary Bundle Block is an SDNV and is therefore variable-length. Two-octet SDNVs are shown here for convenience in representation.

The Creation Timestamp time field in the Primary Bundle Block is an SDNV and is therefore variable-length. A four-octet SDNV is shown here for convenience in representation.

The Creation Timestamp sequence number field in the Primary Bundle Block is an SDNV and is therefore variable-length. A four-octet SDNV is shown here for convenience in representation.

The Lifetime field in the Primary Bundle Block is an SDNV and is therefore variable-length. A four-octet SDNV is shown here for convenience in representation.

The dictionary length field of the Primary Bundle Block is an SDNV and is therefore variable-length. A four-octet SDNV is shown here for convenience in representation.

The fragment offset field of the Primary Bundle Block is present only if the Fragment flag in the block's processing flags byte is set to 1. It is an SDNV and is therefore variable-length; a four-octet SDNV is shown here for convenience in representation.

The total application data unit length field of the Primary Bundle Block is present only if the Fragment flag in the block's processing flags byte is set to 1. It is an SDNV and is therefore variable-length; a four-octet SDNV is shown here for convenience in representation.

The block processing control ("Proc.") flags field of the Payload Block is an SDNV and is therefore variable-length. A one-octet SDNV is shown here for convenience in representation.

The block length field of the Payload Block is an SDNV and is therefore variable-length. A two-octet SDNV is shown here for convenience in representation.

4.5.1. Primary Bundle Block

The primary bundle block contains the basic information needed to route bundles to their destinations. The fields of the primary bundle block are:

Version: A 1-byte field indicating the version of the bundle protocol that constructed this block. The present document describes version 0x05 of the bundle protocol.

Bundle Processing Control Flags: The Bundle Processing Control Flags field is an SDNV that contains the bundle processing control flags discussed in Section 4.2 above.

Block Length: The Block Length field is an SDNV that contains the aggregate length of all remaining fields of the block.

Destination Scheme Offset: The Destination Scheme Offset field contains the offset within the dictionary byte array of the scheme name of the endpoint ID of the bundle's destination, i.e., the endpoint containing the node(s) at which the bundle is to be delivered.

Destination SSP Offset: The Destination SSP Offset field contains the offset within the dictionary byte array of the scheme-specific part of the endpoint ID of the bundle's destination.

Source Scheme Offset: The Source Scheme Offset field contains the offset within the dictionary byte array of the scheme name of the endpoint ID of the bundle's nominal source, i.e., the endpoint nominally containing the node from which the bundle was initially transmitted.

Source SSP Offset: The Source SSP Offset field contains the offset within the dictionary byte array of the scheme-specific part of the endpoint ID of the bundle's nominal source.

Report-to Scheme Offset: The Report-to Scheme Offset field contains the offset within the dictionary byte array of the scheme name of the ID of the endpoint to which status reports pertaining to the forwarding and delivery of this bundle are to be transmitted.

Report-to SSP Offset: The Report-to SSP Offset field contains the offset within the dictionary byte array of the scheme-specific part of the ID of the endpoint to which status reports pertaining to the forwarding and delivery of this bundle are to be transmitted.

Custodian Scheme Offset: The "current custodian endpoint ID" of a primary bundle block identifies an endpoint whose membership includes the node that most recently accepted custody of the bundle upon forwarding this bundle. The Custodian Scheme Offset field contains the offset within the dictionary byte array of the scheme name of the current custodian endpoint ID.

Custodian SSP Offset: The Destination SSP Offset field contains the offset within the dictionary byte array of the scheme-specific part of the current custodian endpoint ID.

Creation Timestamp: The creation timestamp is a pair of SDNVs that, together with the source endpoint ID and (if the bundle is a fragment) the fragment offset and payload length, serve to identify the bundle. The first SDNV of the timestamp is the bundle's creation time while the second is the bundle's creation timestamp sequence number. Bundle creation time is the time - expressed in seconds since the start of the year 2000, on the Coordinated Universal Time (UTC) scale [UTC] - at which the transmission request was received that resulted in the creation of the bundle. Sequence count is the latest value (as of the time at which that transmission request was received) of a monotonically increasing positive integer counter managed by the source node's bundle protocol agent that may be reset to zero whenever the current time advances by one second. A source Bundle Protocol Agent must never create two distinct bundles with the same source endpoint ID and bundle creation timestamp. The combination of source endpoint ID and bundle creation timestamp therefore serves to identify a single transmission request, enabling it to be acknowledged by the receiving application (provided the source endpoint ID is not "dtn:none").

Lifetime: The lifetime field is an SDNV that indicates the time at which the bundle's payload will no longer be useful, encoded as a number of seconds past the creation time. When the current time is greater than the creation time plus the lifetime, bundle nodes need no longer retain or forward the bundle; the bundle may be deleted from the network.

Dictionary Length: The Dictionary Length field is an SDNV that contains the length of the dictionary byte array.

Dictionary: The Dictionary field is an array of bytes formed by concatenating the null-terminated scheme names and SSPs of all endpoint IDs referenced by any fields in this Primary Block together with, potentially, other endpoint IDs referenced by fields in other TBD DTN protocol blocks. Its length is given by the value of the Dictionary Length field.

Fragment Offset: If the Bundle Processing Control Flags of this Primary block indicate that the bundle is a fragment, then the Fragment Offset field is an SDNV indicating the offset from the start of the original application data unit at which the bytes comprising the payload of this bundle were located. If not, then the Fragment Offset field is omitted from the block.

Total Application Data Unit Length: If the Bundle Processing Control Flags of this Primary block indicate that the bundle is a fragment, then the Total Application Data Unit Length field is an SDNV indicating the total length of the original application data unit of which this bundle's payload is a part. If not, then the Total Application Data Unit Length field is omitted from the block.

4.5.2. Canonical Bundle Block Format

Every bundle block of every type other than the primary bundle block comprises the following fields, in this order:

- Block type code, expressed as an 8-bit unsigned binary integer. Bundle block type code 1 indicates that the block is a bundle payload block. Block type codes 192 through 255 are not defined in this specification and are available for private and/or experimental use. All other values of the block type code are reserved for future use.

- Block processing control flags, an unsigned integer expressed as an SDNV. The individual bits of this integer are used to invoke selected block processing control features.

- Block EID reference count and EID references (optional). If and only if the block references EID elements in the primary block's dictionary, the 'block contains an EID-reference field' flag in the block processing control flags is set to 1 and the block includes an EID reference field consisting of a count of EID references expressed as an SDNV followed by the EID references themselves. Each EID reference is a pair of SDNVs. The first SDNV of each EID reference contains the offset of a scheme name in the primary block's dictionary, and the second SDNV of each reference contains the offset of a scheme-specific part in the dictionary.

- Block data length, an unsigned integer expressed as an SDNV. The Block data length field contains the aggregate length of all remaining fields of the block, i.e., the block-type-specific data fields.

- Block-type-specific data fields, whose format and order are type-specific and whose aggregate length in octets is the value of the block data length field. All multi-byte block-type-specific data fields are represented in network byte order.

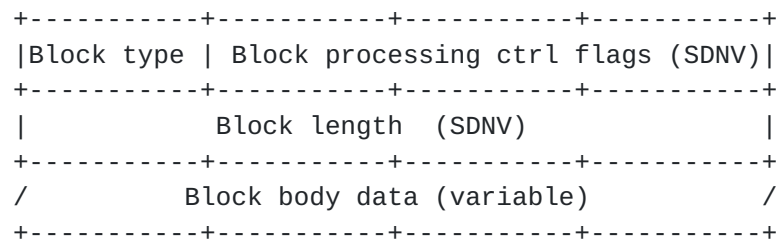


Figure 6: Block layout without EID reference list.

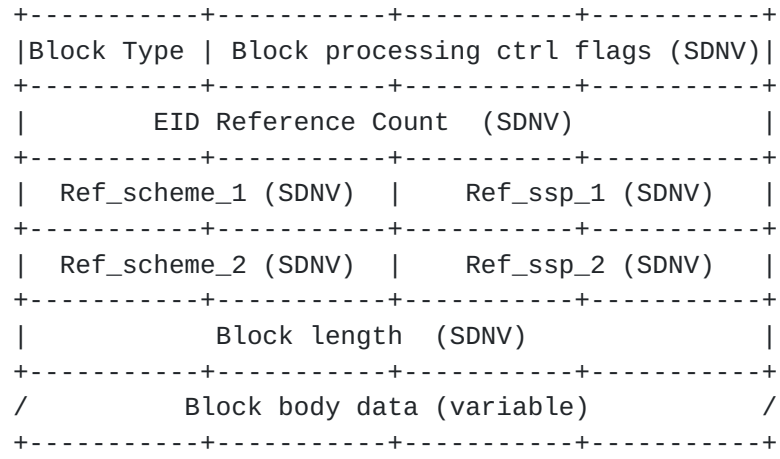


Figure 7: Block layout showing two EID references.

4.5.3. Bundle Payload Block

The fields of the bundle payload block are:

Block Type: The Block Type field is a 1-byte field that indicates the type of the block. For the bundle payload block this field contains the value 1.

Block Processing Control Flags: The Block Processing Control Flags field is an SDNV that contains the block processing control flags discussed in Section 4.3 above.

Block Length: The Block Length field is an SDNV that contains the aggregate length of all remaining fields of the block - which is to say, the length of the bundle's payload.

Payload: The application data carried by this bundle.

That is, bundle payload blocks follow the canonical format of the previous section with the restriction that the 'block contains an EID-reference field' bit of the block processing control flags is

never set. The block body data for payload blocks is the application data carried by the bundle.

4.6. Extension Blocks

"Extension blocks" are all blocks other than the primary and payload blocks. Because extension blocks are not defined in the Bundle Protocol specification (the present document), not all nodes conforming to this specification will necessarily instantiate Bundle Protocol implementations that include procedures for processing (that is, recognizing, parsing, acting on, and/or producing) all extension blocks. It is therefore possible for a node to receive a bundle that includes extension blocks which the node cannot process.

Whenever a bundle is forwarded that contains one or more extension blocks that could not be processed, the "Block was forwarded without being processed" flag must be set to 1 within the block processing flags of each such block. For each block flagged in this way, the flag may optionally be cleared (i.e., set to zero) by another node that subsequently receives the bundle and is able to process that block; the specifications defining the various extension blocks are expected to define the circumstances under which this flag may be cleared, if any.

4.7. Dictionary revision

Any strings (scheme names and SSPs) in a bundle's dictionary that are not referenced from the bundle's primary block nor from the block EID reference field of any extension block may be removed from the dictionary at the time the bundle is forwarded.

Whenever removal of a string from the dictionary causes the offsets (within the dictionary byte array) of any other strings to change, all endpoint ID references that refer to those strings must be adjusted at the same time. Note that these references may be in the primary block and/or in the block EID reference fields of extension blocks.

5. Bundle Processing

The bundle processing procedures mandated in this section and in Section 6 govern the operation of the Bundle Protocol Agent and the Application Agent administrative element of each bundle node. They are neither exhaustive nor exclusive. That is, supplementary DTN protocol specifications (including, but not restricted to, the Bundle Security Protocol [BSP]) may require that additional measures be taken at specified junctures in these procedures. Such additional measures shall not override or supersede the mandated bundle protocol procedures, except that they may in some cases make these procedures moot by requiring, for example, that implementations conforming to the supplementary protocol terminate the processing of a given incoming or outgoing bundle due to a fault condition recognized by that protocol.

5.1. Generation of administrative records

All initial transmission of bundles is in response to bundle transmission requests presented by nodes' application agents. When required to "generate" an administrative record (a bundle status report or a custody signal), the bundle protocol agent itself is responsible for causing a new bundle to be transmitted, conveying that record. In concept, the bundle protocol agent discharges this responsibility by directing the administrative element of the node's application agent to construct the record and request its transmission as detailed in Section 6 below; in practice, the manner in which administrative record generation is accomplished is an implementation matter, provided the constraints noted in Section 6 are observed.

Under some circumstances the requesting of status reports could result in an unacceptable increase in the bundle traffic in the network. For this reason, the generation of status reports is mandatory only in one case, the deletion of a bundle for which custody transfer is requested. In all other cases the decision on whether or not to generate a requested status report is left to the discretion of the bundle protocol agent. Mechanisms that could assist in making such decisions, such as pre-placed agreements authorizing the generation of status reports under specified circumstances, are beyond the scope of this specification.

Notes on administrative record terminology:

- A "bundle reception status report" is a bundle status report with the "reporting node received bundle" flag set to 1.

- A "custody acceptance status report" is a bundle status report with the "reporting node accepted custody of bundle" flag set to 1.

- A "bundle forwarding status report" is a bundle status report with the "reporting node forwarded the bundle" flag set to 1.

- A "bundle delivery status report" is a bundle status report with the "reporting node delivered the bundle" flag set to 1.

- A "bundle deletion status report" is a bundle status report with the "reporting node deleted the bundle" flag set to 1.

- A "Succeeded" custody signal is a custody signal with the "custody transfer succeeded" flag set to 1.

- A "Failed" custody signal is a custody signal with the "custody transfer succeeded" flag set to zero.

- The "current custodian" of a bundle is the endpoint identified by the current custodian endpoint ID in the bundle's primary block.

5.2. Bundle Transmission

The steps in processing a bundle transmission request are:

Step 1: If custody transfer is requested for this bundle transmission and, moreover, custody acceptance by the source node is required, then either the bundle protocol agent must commit to accepting custody of the bundle - in which case processing proceeds from Step 2 - or else the request cannot be honored and all remaining steps of this procedure must be skipped. The bundle protocol agent must not commit to accepting custody of a bundle if the conditions under which custody of the bundle may be accepted are not satisfied. The conditions under which a node may accept custody of a bundle whose destination is not a singleton endpoint are not defined in this specification.

Step 2: Transmission of the bundle is initiated. An outbound bundle must be created per the parameters of the bundle transmission request, with current custodian endpoint ID set to the null endpoint ID "dtn:none" and with the retention constraint "Dispatch pending". The source endpoint ID of the bundle must be either the ID of an endpoint of which the node is a member or else the null endpoint ID "dtn:none".

Step 3: Processing proceeds from Step 1 of Section 5.4.

5.3. Bundle Dispatching

The steps in dispatching a bundle are:

Step 1: If the bundle's destination endpoint is an endpoint of which the node is a member, the bundle delivery procedure defined in Section 5.7 must be followed.

Step 2: Processing proceeds from Step 1 of Section 5.4.

5.4. Bundle Forwarding

The steps in forwarding a bundle are:

Step 1: The retention constraint "Forward pending" must be added to the bundle, and the bundle's "Dispatch pending" retention constraint must be removed.

Step 2: The bundle protocol agent must determine whether or not forwarding is contraindicated for any of the reasons listed in Figure 12. In particular:

* The bundle protocol agent must determine which endpoint(s) to forward the bundle to. The bundle protocol agent may choose either to forward the bundle directly to its destination endpoint (if possible) or else to forward the bundle to some other endpoint(s) for further forwarding. The manner in which this decision is made may depend on the scheme name in the destination endpoint ID but in any case is beyond the scope of this document. If the agent finds it impossible to select any endpoint(s) to forward the bundle to, then forwarding is contraindicated.

* Provided the bundle protocol agent succeeded in selecting the endpoint(s) to forward the bundle to, the bundle protocol agent must select the convergence layer adapter(s) whose services will enable the node to send the bundle to the nodes of the minimum reception group of each selected endpoint. The manner in which the appropriate convergence layer adapters are selected may depend on the scheme name in the destination endpoint ID but in any case is beyond the scope of this document. If the agent finds it impossible to select convergence layer adapters to use in forwarding this bundle, then forwarding is contraindicated.

Step 3: If forwarding of the bundle is determined to be contraindicated for any of the reasons listed in Figure 12, then the Forwarding Contraindicated procedure defined in Section 5.4.1 must be followed; the remaining steps of Section 5 are skipped at this time.

Step 4: If the bundle's custody transfer requested flag (in the bundle processing flags field) is set to 1 then the custody transfer procedure defined in Section 5.10.2 must be followed.

Step 5: For each endpoint selected for forwarding, the bundle protocol agent must invoke the services of the selected convergence layer adapter(s) in order to effect the sending of the bundle to the nodes constituting the minimum reception group of that endpoint. Determining the time at which the bundle is to be sent by each convergence layer adapter is an implementation matter.

To keep from possibly invalidating bundle security, the sequencing of the blocks in a forwarded bundle must not be changed as it transits a node; received blocks must be transmitted in the same relative order as that in which they were received. While blocks may be added to bundles as they transit intermediate nodes, removal of blocks that do not have their 'Discard block if it can't be processed' flag in the block processing control flags set to 1 may cause security to fail.

Step 6: When all selected convergence layer adapters have informed the bundle protocol agent that they have concluded their data sending procedures with regard to this bundle:

* If the "request reporting of bundle forwarding" flag in the bundle's status report request field is set to 1, then a bundle forwarding status report should be generated, destined for the bundle's report-to endpoint ID. If the bundle has the retention constraint "custody accepted" and all of the nodes in the minimum reception group of the endpoint selected for forwarding are known to be unable to send bundles back to this node, then the reason code on this bundle forwarding status report must be "forwarded over unidirectional link"; otherwise the reason code must be "no additional information".

* The bundle's "Forward pending" retention constraint must be removed.

5.4.1. Forwarding Contraindicated

The steps in responding to contraindication of forwarding for some reason are:

Step 1: The bundle protocol agent must determine whether or not to declare failure in forwarding the bundle for this reason. Note: this decision is likely to be influenced by the reason for which forwarding is contraindicated.

Step 2: If forwarding failure is declared, then the Forwarding Failed procedure defined in 4.4.2 must be followed. Otherwise, (a) if the bundle's custody transfer requested flag (in the bundle processing flags field) is set to 1 then the custody transfer procedure defined in Section 5.10 must be followed; (b) when - at some future time - the forwarding of this bundle ceases to be contraindicated, processing proceeds from Step 5 of Section 5.4.

5.4.2. Forwarding Failed

The steps in responding to a declaration of forwarding failure for some reason are:

Step 1: If the bundle's custody transfer requested flag (in the bundle processing flags field) is set to 1, custody transfer failure must be handled. Procedures for handling failure of custody transfer for a bundle whose destination is not a singleton endpoint are not defined in this specification. For a bundle whose destination is a singleton endpoint, the bundle protocol agent must handle the custody transfer failure by generating a "Failed" custody signal for the bundle, destined for the bundle's current custodian; the custody signal must contain a reason code corresponding to the reason for which forwarding was determined to be contraindicated. (Note that discarding the bundle will not delete it from the network, since the current custodian still has a copy.)

Step 2: If the bundle's destination endpoint is an endpoint of which the node is a member, then the bundle's "Forward pending" retention constraint must be removed. Otherwise the bundle must be deleted: the bundle deletion procedure defined in 4.13 must be followed, citing the reason for which forwarding was determined to be contraindicated.

5.5. Bundle Expiration

A bundle expires when the current time is greater than the bundle's creation time plus its lifetime as specified in the primary bundle block. Bundle expiration may occur at any point in the processing of a bundle. When a bundle expires, the bundle protocol agent must delete the bundle for the reason "lifetime expired": the bundle deletion procedure defined in Section 5.13 must be followed.

5.6. Bundle Reception

The steps in processing a bundle received from another node are:

Step 1: The retention constraint "Dispatch pending" must be added to the bundle.

Step 2: If the "request reporting of bundle reception" flag in the bundle's status report request field is set to 1, then a bundle reception status report with reason code "No additional information" should be generated, destined for the bundle's report-to endpoint ID.

Step 3: For each block in the bundle that is an extension block that the bundle protocol agent cannot process:

* If the block processing flags in that block indicate that a status report is requested in this event, then a bundle reception status report with reason code "Block unintelligible" should be generated, destined for the bundle's report-to endpoint ID.

* If the block processing flags in that block indicate that the bundle must be deleted in this event, then the bundle protocol agent must delete the bundle for the reason "Block unintelligible"; the bundle deletion procedure defined in 4.13 must be followed and all remaining steps of the bundle reception procedure must be skipped.

* If the block processing flags in that block do NOT indicate that the bundle must be deleted in this event but do indicate that the block must be discarded, then the bundle protocol agent must remove this block from the bundle.

* If the block processing flags in that block indicate NEITHER that the bundle must be deleted NOR that the block must be discarded, then the bundle protocol agent must set to 1 the "Block was forwarded without being processed" flag in the block processing flags of the block.

Step 4: If the bundle's custody transfer requested flag (in the bundle processing flags field) is set to 1 and the bundle has the same source endpoint ID, creation timestamp, and (if the bundle is a fragment) fragment offset and payload length as another bundle that (a) has not been discarded and (b) currently has the retention constraint "Custody accepted", custody transfer redundancy must be handled; otherwise, processing proceeds from Step 5. Procedures for handling redundancy in custody transfer for a bundle whose destination is not a singleton endpoint are not defined in this specification. For a bundle whose destination is a singleton endpoint, the bundle protocol agent must handle custody transfer redundancy by generating a "Failed" custody signal for this bundle with reason code "Redundant reception", destined for this bundle's current custodian, and removing this bundle's "Dispatch pending" retention constraint.

Step 5: Processing proceeds from Step 1 of Section 5.3.

5.7. Local Bundle Delivery

The steps in processing a bundle that is destined for an endpoint of which this node is a member are:

Step 1: If the received bundle is a fragment, the application data unit reassembly procedure described in Section 5.9 must be followed. If this procedure results in reassembly of the entire original application data unit, processing of this bundle (whose fragmentary payload has been replaced by the reassembled application data unit) proceeds from Step 2; otherwise the retention constraint "Reassembly pending" must be added to the bundle and all remaining steps of this procedure are skipped.

Step 2: Delivery depends on the state of the registration whose endpoint ID matches that of the destination of the bundle:

* If the registration is in the Active state, then the bundle must be delivered subject to this registration (see Section 3.1 above) as soon as all previously received bundles that are deliverable subject to this registration have been delivered.

* If the registration is in the Passive state, then the registration's delivery failure action must be taken (see Section 3.1 above).

Step 3: As soon as the bundle has been delivered:

* If the "request reporting of bundle delivery" flag in the bundle's status report request field is set to 1, then a bundle delivery status report should be generated, destined for the bundle's report-to endpoint ID. Note that this status report only states that the payload has been delivered to the application agent, not that the application agent has processed that payload.

* If the bundle's custody transfer requested flag (in the bundle processing flags field) is set to 1, custodial delivery must be reported. Procedures for reporting custodial delivery for a bundle whose destination is not a singleton endpoint are not defined in this specification. For a bundle whose destination is a singleton endpoint, the bundle protocol agent must report custodial delivery by generating a "Succeeded" custody signal for the bundle, destined for the bundle's current custodian.

5.8. Bundle Fragmentation

It may at times be necessary for bundle protocol agents to reduce the sizes of bundles in order to forward them. This might be the case, for example, if the endpoint to which a bundle is to be forwarded is accessible only via intermittent contacts and no upcoming contact is long enough to enable the forwarding of the entire bundle.

The size of a bundle can be reduced by "fragmenting" the bundle. To fragment a bundle whose payload is of size M is to replace it with two "fragments" - new bundles with the same source endpoint ID and creation timestamp as the original bundle - whose payloads are the first N and the last (M - N) bytes of the original bundle's payload, where 0 < N < M. Note that fragments may themselves be fragmented, so fragmentation may in effect replace the original bundle with more than two fragments. (However, there is only one 'level' of fragmentation, as in IP fragmentation.)

Any bundle whose primary block's bundle processing flags do NOT indicate that it must not be fragmented may be fragmented at any time, for any purpose, at the discretion of the bundle protocol agent.

Fragmentation shall be constrained as follows:

- The concatenation of the payloads of all fragments produced by fragmentation must always be identical to the payload of the bundle that was fragmented. Note that the payloads of fragments resulting from different fragmentation episodes, in different parts of the network, may be overlapping subsets of the original bundle's payload.

- The bundle processing flags in the primary block of each fragment must be modified to indicate that the bundle is a fragment, and both fragment offset and total application data unit length must be provided at the end of each fragment's primary bundle block.

- The primary blocks of the fragments will differ from that of the fragmented bundle as noted above.

- The payload blocks of fragments will differ from that of the fragmented bundle as noted above.

- All blocks that precede the payload block at the time of fragmentation must be replicated in the fragment with the lowest offset.

- All blocks that follow the payload block at the time of fragmentation must be replicated in the fragment with the highest offset.

- If the 'Block must be replicated in every fragment' bit is set to one then the block must be replicated in every fragment.

- If the 'Block must be replicated in every fragment' bit is set to zero, the block should be replicated in only one fragment.

- The relative order of all blocks that are present in a fragment must be the same as in the bundle prior to fragmentation.

5.9. Application Data Unit Reassembly

If the concatenation - as informed by fragment offsets and payload lengths - of the payloads of all previously received fragments with the same source endpoint ID and creation timestamp as this fragment, together with the payload of this fragment, forms a byte array whose length is equal to the total application data unit length in the fragment's primary block, then:

- This byte array - the reassembled application data unit - must replace the payload of this fragment.

- The "Reassembly pending" retention constraint must be removed from every other fragment whose payload is a subset of the reassembled application data unit.

Note: reassembly of application data units from fragments occurs at destination endpoints as necessary; an application data unit may also be reassembled at some other endpoint on the route to the destination.

5.10. Custody Transfer

The conditions under which a node may accept custody of a bundle whose destination is not a singleton endpoint are not defined in this specification.

The decision as to whether or not to accept custody of a bundle whose destination is a singleton endpoint is an implementation matter which may involve both resource and policy considerations; however, if the bundle protocol agent has committed to accepting custody of the bundle (as described in Step 1 of Section 5.2) then custody must be accepted.

If the bundle protocol agent elects to accept custody of the bundle, then it must follow the custody acceptance procedure defined in Section 5.10.1.

5.10.1. Custody Acceptance

Procedures for acceptance of custody of a bundle whose destination is not a singleton endpoint are not defined in this specification.

Procedures for acceptance of custody of a bundle whose destination is a singleton endpoint are defined as follows.

The retention constraint "Custody accepted" must be added to the bundle.

If the "request custody acceptance reporting" flag in the bundle's status report request field is set to 1, a custody acceptance status report should be generated, destined for the report-to endpoint ID of the bundle. However, if a bundle reception status report was generated for this bundle (step 1 of Section 5.6) then this report should be generated by simply turning on the "Reporting node accepted custody of bundle" flag in that earlier report's status flags byte.

The bundle protocol agent must generate a "Succeeded" custody signal for the bundle, destined for the bundle's current custodian.

The bundle protocol agent must assert the new current custodian for the bundle. It does so by changing the current custodian endpoint ID in the bundle's primary block to the endpoint ID of one of the singleton endpoints in which the node is registered. This may entail appending that endpoint ID's null-terminated scheme name and SSP to the dictionary byte array in the bundle's primary block, and in some case it may also enable the (optional) removal of the current custodian endpoint ID's scheme name and/or SSP from the dictionary.

The bundle protocol agent may set a custody transfer countdown timer for this bundle; upon expiration of this timer prior to expiration of the bundle itself and prior to custody transfer success for this bundle, the custody transfer failure procedure detailed in section Section 5.12 must be followed. The manner in which the countdown interval for such a timer is determined is an implementation matter.

The bundle should be retained in persistent storage if possible.

5.10.2. Custody Release

Procedures for release of custody of a bundle whose destination is not a singleton endpoint are not defined in this specification.

When custody of a bundle is released, where the destination of the bundle is a singleton endpoint, the "Custody accepted" retention constraint must be removed from the bundle and any custody transfer timer that has been established for this bundle must be destroyed.

5.11. Custody Transfer Success

Procedures for determining custody transfer success for a bundle whose destination is not a singleton endpoint are not defined in this specification.

Upon receipt of a "Succeeded" custody signal at a node that is a custodial node of the bundle identified in the custody signal, where the destination of the bundle is a singleton endpoint, custody of the bundle must be released as described in Section 5.10.2.

5.12. Custody Transfer Failure

Procedures for determining custody transfer failure for a bundle whose destination is not a singleton endpoint are not defined in this specification. Custody transfer for a bundle whose destination is a singleton endpoint is determined to have failed at a custodial node for that bundle when either (a) that node's custody transfer timer for that bundle (if any) expires or (b) a "Failed" custody signal for that bundle is received at that node.

Upon determination of custody transfer failure, the action taken by the bundle protocol agent is implementation-specific and may depend on the nature of the failure. For example, if custody transfer failure was inferred from expiration of a custody transfer timer or was asserted by a "Failed" custody signal with the "Depleted storage" reason code, the bundle protocol agent might choose to re-forward the bundle, possibly on a different route (Section 5.4). Receipt of a "Failed" custody signal with the "Redundant reception" reason code, on the other hand, might cause the bundle protocol agent to release custody of the bundle and to revise its algorithm for computing countdown intervals for custody transfer timers.

5.13. Bundle Deletion

The steps in deleting a bundle are:

Step 1: If the retention constraint "Custody accepted" currently prevents this bundle from being discarded, and the destination of the bundle is a singleton endpoint, then:

* Custody of the node is released as described in Section 5.10.2.

* A bundle deletion status report citing the reason for deletion must be generated, destined for the bundle's report-to endpoint ID.

Otherwise, if the "request reporting of bundle deletion" flag in the bundle's status report request field is set to 1, then a bundle deletion status report citing the reason for deletion should be generated, destined for the bundle's report-to endpoint ID.

Step 2: All of the bundle's retention constraints must be removed.

5.14. Discarding a Bundle

As soon as a bundle has no remaining retention constraints it may be discarded.

5.15. Canceling a Transmission

When requested to cancel a specified transmission, where the bundle created upon initiation of the indicated transmission has not yet been discarded, the bundle protocol agent must delete that bundle for the reason "transmission canceled". For this purpose, the procedure defined in Section 5.13 must be followed.

5.16. Polling

When requested to poll a specified registration that is in Passive state, the bundle protocol agent must immediately deliver the least recently received bundle that is deliverable subject to the indicated registration, if any.

6. Administrative Record Processing

6.1. Administrative Records

Administrative records are standard application data units that are used in providing some of the features of the Bundle Protocol. Two types of administrative records have been defined to date: bundle status reports and custody signals.

Every administrative record consists of a four-bit record type code followed by four bits of administrative record flags, followed by record content in type-specific format. Record type codes are defined as follows:

Value	Meaning
0001	Bundle status report.
0010	Custody signal.
(other)	Reserved for future use.

Figure 8: Administrative Record Type Codes.

Value	Meaning
0001	Record is for a fragment; fragment offset and length fields are present.
(other)	Reserved for future use.

Figure 9: Administrative Record Flags.

All time values in administrative records are UTC times expressed in "DTN time" representation. A DTN time consists of an SDNV indicating the number of seconds since the start of the year 2000, followed by an SDNV indicating the number of nanoseconds since the start of the indicated second.

The contents of the various types of administrative records are described below.

6.1.1. Bundle Status Reports

The transmission of 'bundle status reports' under specified conditions is an option that can be invoked when transmission of a bundle is requested. These reports are intended to provide information about how bundles are progressing through the system, including notices of receipt, custody transfer, forwarding, final delivery, and deletion. They are transmitted to the Report-to endpoints of bundles.

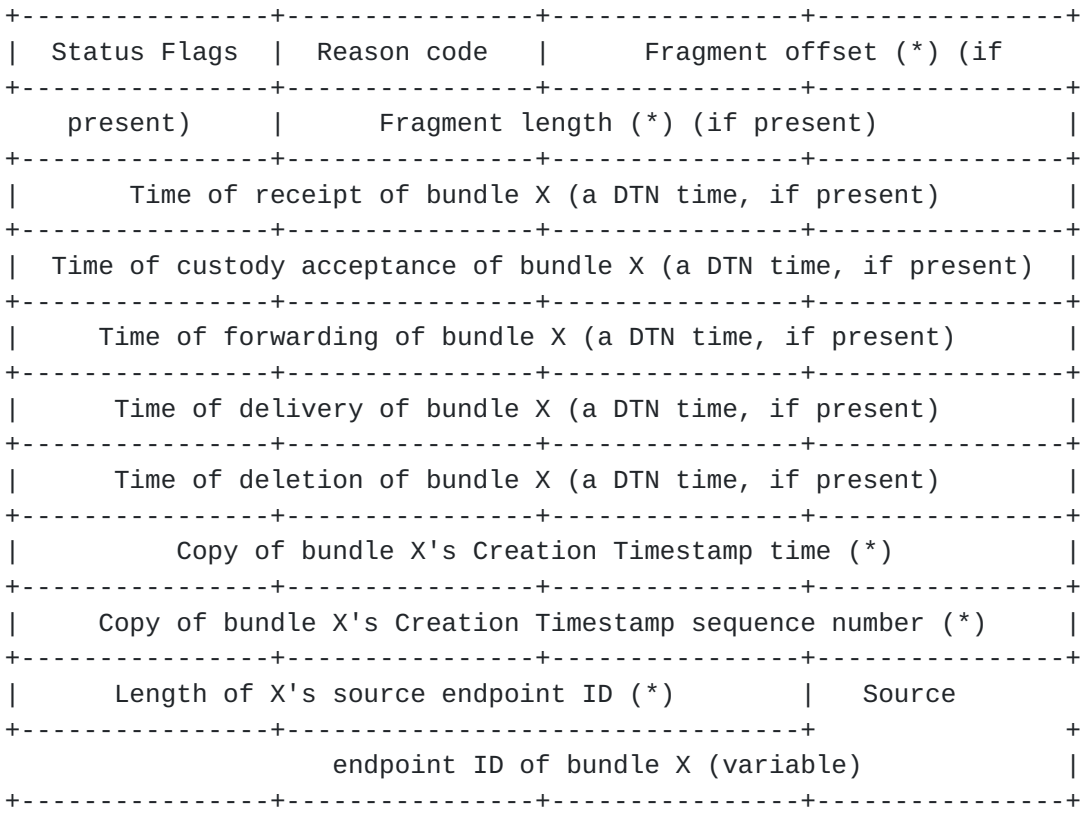


Figure 10: Bundle status report format

(*) Notes:

The Fragment Offset field, if present, is an SDNV and is therefore variable-length. A three-octet SDNV is shown here for convenience in representation.

The Fragment Length field, if present, is an SDNV and is therefore variable-length. A three-octet SDNV is shown here for convenience in representation.

The Creation Timestamp fields replicate the Creation Timestamp fields

in the primary block of the subject bundle. As such they are SDNVs (see 3.5.1 above) and are therefore variable-length. Four-octet SDNVs are shown here for convenience in representation.

The source endpoint ID length field is an SDNV and is therefore variable-length. A three-octet SDNV is shown here for convenience in representation.

The fields in a bundle status report are:

Status Flags: A 1-byte field containing the following flags:

Value	Meaning
00000001	Reporting node received bundle.
00000010	Reporting node accepted custody of bundle.
00000100	Reporting node forwarded the bundle.
00001000	Reporting node delivered the bundle.
00010000	Reporting node deleted the bundle.
00100000	Unused.
01000000	Unused.
10000000	Unused.

Figure 11: Status Flags for Bundle Status Reports

Reason Code: A 1-byte field explaining the value of the flags in the status flags byte. The list of status report reason codes provided here is neither exhaustive nor exclusive; supplementary DTN protocol specifications (including, but not restricted to, the Bundle Security Protocol [BSP]) may define additional reason codes. Status report reason codes are defined as follows:

Value	Meaning
0x00	No additional information.
0x01	Lifetime expired.
0x02	Forwarded over unidirectional link.
0x03	Transmission canceled.
0x04	Depleted storage.
0x05	Destination endpoint ID unintelligible.
0x06	No known route to destination from here.
0x07	No timely contact with next node on route.
0x08	Block unintelligible.
(other)	Reserved for future use.

Figure 12: Status Report Reason Codes

Fragment Offset: If the bundle fragment bit is set in the status flags, then the offset (within the original application data unit) of the payload of the bundle that caused the status report to be generated is included here.

Fragment length: If the bundle fragment bit is set in the status flags, then the length of the payload of the subject bundle is included here.

Time of Receipt (if present): If the bundle-received bit is set in the status flags, then a DTN time indicating the time at which the bundle was received at the reporting node is included here.

Time of Custody Acceptance (if present): If the custody-accepted bit is set in the status flags, then a DTN time indicating the time at which custody was accepted at the reporting node is included here.

Time of Forward (if present): If the bundle-forwarded bit is set in the status flags, then a DTN time indicating the time at which the bundle was first forwarded at the reporting node is included here.

Time of Delivery (if present): If the bundle-delivered bit is set in the status flags, then a DTN time indicating the time at which the bundle was delivered at the reporting node is included here.

Time of Deletion (if present): If the bundle-deleted bit is set in the status flags, then a DTN time indicating the time at which the bundle was deleted at the reporting node is included here.

Creation Timestamp of Subject Bundle: A copy of the creation timestamp of the bundle that caused the status report to be generated.

Length of Source Endpoint ID: The length in bytes of the source endpoint ID of the bundle that caused the status report to be generated.

Source Endpoint ID text: The text of the source endpoint ID of the bundle that caused the status report to be generated.

6.1.2. Custody Signals

Custody signals are administrative records that effect custody transfer operations. They are transmitted to the endpoints that are the current custodians of bundles.

Custody signals have the following format.

Custody Signal regarding bundle 'X':

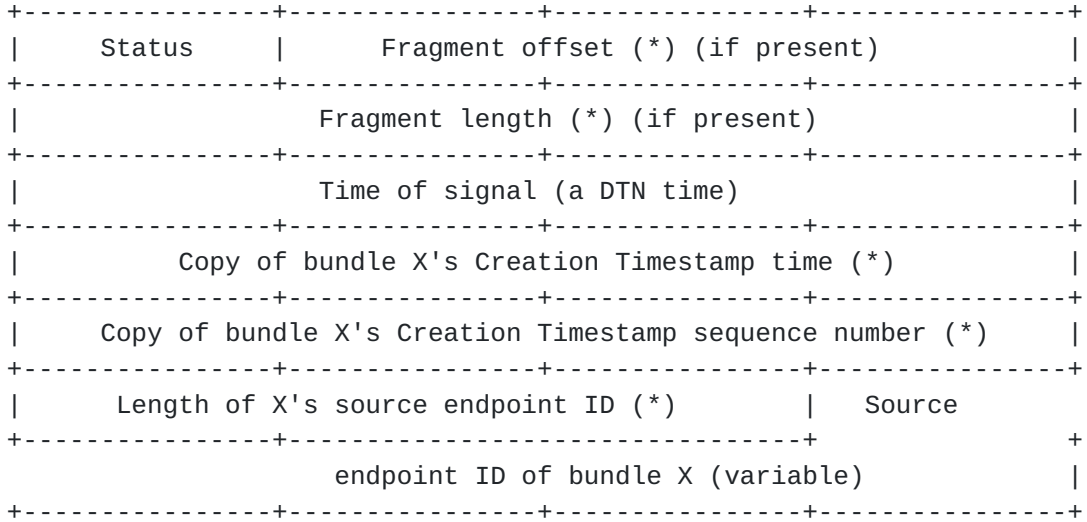


Figure 13: Custody signal format.

(*) Notes:

The Fragment Offset field, if present, is an SDNV and is therefore variable-length. A three-octet SDNV is shown here for convenience in representation.

The Fragment Length field, if present, is an SDNV and is therefore variable-length. A four-octet SDNV is shown here for convenience in representation.

The Creation Timestamp fields replicate the Creation Timestamp fields in the primary block of the subject bundle. As such they are SDNVs (see Section 4.5.1 above) and are therefore variable-length. Four-octet SDNVs are shown here for convenience in representation.

The source endpoint ID length field is an SDNV and is therefore variable-length. A three-octet SDNV is shown here for convenience in representation.

The fields in a custody signal are:

Status: A 1-byte field containing a 1-bit "custody transfer succeeded" flag followed by a 7-bit reason code explaining the value of that flag. Custody signal reason codes are defined as follows:

Value	Meaning
0x00	No additional information.
0x01	Reserved for future use.
0x02	Reserved for future use.
0x03	Redundant reception (reception by a node that is a custodial node for this bundle).
0x04	Depleted storage.
0x05	Destination endpoint ID unintelligible.
0x06	No known route to destination from here.
0x07	No timely contact with next node on route.
0x08	Block unintelligible.
(other)	Reserved for future use.

Figure 14: Custody Signal Reason Codes

Fragment offset: If the bundle fragment bit is set in the status flags, then the offset (within the original application data unit) of the payload of the bundle that caused the status report to be generated is included here.

Fragment length: If the bundle fragment bit is set in the status flags, then the length of the payload of the subject bundle is included here.

Time of Signal: A DTN time indicating the time at which the signal was generated.

Creation Timestamp of Subject Bundle: A copy of the creation timestamp of the bundle to which the signal applies.

Length of Source Endpoint ID: The length in bytes of the source endpoint ID of the bundle to which the signal applied.

Source Endpoint ID text: The text of the source endpoint ID of the bundle to which the signal applies.

6.2. Generation of Administrative Records

Whenever the application agent's administrative element is directed by the bundle protocol agent to generate an administrative record with reference to some bundle, the following procedure must be followed:

Step 1: The administrative record must be constructed. If the referenced bundle is a fragment, the administrative record must have the Fragment flag set and must contain the fragment offset and fragment length fields; the value of the fragment offset field must be the value of the referenced bundle's fragment offset, and the value of the fragment length field must be the length of the referenced bundle's payload.

Step 2: A request for transmission of a bundle whose payload is this administrative record must be presented to the bundle protocol agent.

6.3. Reception of Custody Signals

For each received custody signal that has the Custody Transfer Succeeded flag set to 1, the administrative element of the application agent must direct the bundle protocol agent to follow the custody transfer success procedure in Section 5.11.

For each received custody signal that has the Custody Transfer Succeeded flag set to 0, the administrative element of the application agent must direct the bundle protocol agent to follow the custody transfer failure procedure in Section 5.12.

7. Services Required of the Convergence Layer

7.1. The Convergence Layer

The successful operation of the end-to-end bundle protocol depends on the operation of underlying protocols at what is termed the "convergence layer"; these protocols accomplish communication between nodes. A wide variety of protocols may serve this purpose, so long as each convergence layer protocol adapter provides a defined minimal set of services to the bundle protocol agent. This convergence layer service specification enumerates those services.

7.2. Summary of Convergence Layer Services

Each convergence layer protocol adapter is expected to provide the following services to the bundle protocol agent:

- sending a bundle to all bundle nodes in the minimum reception group of the endpoint identified by a specified endpoint ID that are reachable via the convergence layer protocol;

- delivering to the bundle protocol agent a bundle that was sent by a remote bundle node via the convergence layer protocol.

The convergence layer service interface specified here is neither exhaustive nor exclusive. That is, supplementary DTN protocol specifications (including, but not restricted to, the Bundle Security Protocol [BSP]) may expect convergence layer adapters which serve BP implementations conforming to those protocols to provide additional services.

8. Security Considerations

The bundle protocol has taken security into concern from the outset of its design. It was always assumed that security services would be needed in the use of the bundle protocol. As a result, the bundle protocol security architecture and the available security services are specified in an accompanying document, the Bundle Security Protocol specification [BSP]; an informative overview of this architecture is provided in [SECO].

The bundle protocol has been designed with the notion that it will be run over networks with scarce resources. For example, the networks might have limited bandwidth, limited connectivity, constrained storage in relay nodes, etc. Therefore, the bundle protocol must ensure that only those entities authorized to send bundles over such constrained environments are actually allowed to do so. All unauthorized entities should be prevented from consuming valuable resources.

Likewise, because of the potentially long latencies and delays involved in the networks that make use of the bundle protocol, data sources should be concerned with the integrity of the data received at the intended destination(s) and may also be concerned with ensuring confidentiality of the data as it traverses the network. Without integrity, the bundle payload data might be corrupted while in transit without the destination able to detect it. Similarly, the data source can be concerned with ensuring that the data can only be used by those authorized; hence the need for confidentiality.

Internal to the bundle-aware overlay network, the bundle nodes should be concerned with the authenticity of other bundle nodes as well as the preservation of bundle payload data integrity as it is forwarded between bundle nodes.

As a result, bundle security is concerned with the authenticity, integrity, and confidentiality of bundles conveyed among bundle nodes. This is accomplished via the use of three, independent security specific bundle blocks which may be used together to provide multiple bundle security services or independently of one another, depending on perceived security threats, mandated security requirements, and security policies that must be enforced.

The Bundle Authentication Block (BAB) ensures the authenticity and integrity of bundles on a hop-by-hop basis between bundle nodes. The BAB allows each bundle node to verify a bundle's authenticity before processing or forwarding the bundle. In this way, entities that are not authorized to send bundles will have unauthorized transmissions blocked by security-aware bundle nodes.

Additionally, to provide "security-source" to "security-destination" bundle authenticity and integrity, the Payload Security Block (PSB) is used. A "security-source" may not actually be the origination point of the bundle but instead may be the first point along the path that is security-aware and is able to apply security services. For example, an enclave of networked systems may generate bundles but only their gateway may be required and/or able to apply security services. The PSB allows any security-enabled entity along the delivery path, in addition to the "security-destination" (the recipient counterpart to the "security-source"), to ensure the bundle's authenticity.

Finally, to provide payload confidentiality, the use of the Confidentiality Block (CB) is available. The bundle payload may be encrypted to provide "security-source" to "security-destination" payload confidentiality/privacy. The CB indicates the cryptographic algorithm and key IDs that were used to encrypt the payload.

Note that removal of strings from the dictionary at a given point in a bundle's end-to-end path, and attendant adjustment of endpoint ID references in the blocks of that bundle, may make it necessary to re-compute values in one or more of the bundle's security blocks.

Bundle security must not be invalidated by forwarding nodes even though they themselves might not use the Bundle Security Protocol. In particular, the sequencing of the blocks in a forwarded bundle must not be changed as it transits a node; received blocks must be transmitted in the same relative order as that in which they were received. While blocks may be added to bundles as they transit intermediate nodes, removal of blocks that do not have their 'Discard block if it can't be processed' flag in the block processing control flags set to 1 may cause security to fail.

Inclusion of the Bundle Security Protocol in any Bundle Protocol implementation is RECOMMENDED. Use of the Bundle Security Protocol in Bundle Protocol operations is OPTIONAL.

9. IANA Considerations

The "dtn:" URI scheme has been provisionally registered by IANA. See http://www.iana.org/assignments/uri-schemes.html for the latest details.

10. References

10.1. Normative References

[IPR] Bradner, S., "Intellectual Property Rights in IETF Technology", RFC 3979, BCP 79, March 2005.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RGTS] Bradner, S., "IETF Rights in Contributions", RFC 3978, BCP 78, March 2005.

[URI] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, STD 66, January 2005.

[URIREG] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", RFC 4395, BCP 115, February 2006.

10.2. Informative References

[ARCH] V. Cerf et. al., "Delay-Tolerant Network Architecture", RFC 4838, April 2007.

[ASN1] "Abstract Syntax Notation One (ASN.1), "ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)," ITU-T Rec. X.690 (2002) | ISO/IEC 8825-1:2002", 2003.

[BSP] Symington, S., "Bundle Security Protocol Specification, work in progress", Work in progress, draft-irtf-dtnrg-bundle-security-03, October 2007.

[NTP] Mills, D., "Network Time Protocol (Version 3) Specification", RFC 1305, March 1992.

[SECO] Farrell, S., Symington, S., Weiss, H., and P. Lovell, "Delay-Tolerant Networking Security Overview", Work in progress, draft-irtf-dtnrg-sec-overview-03, July 2007.

[SIGC] Fall, K., "A Delay-Tolerant Network Architecture for Challenged Internets", SIGCOMM 2003 .

[TUT] Warthman, F., "Delay-Tolerant Networks (DTNs): A

Tutorial", <http://www.dtnrg.org>.

[UTC] Arias, E. and B. Guinot, ""Coordinated universal time UTC: historical background and perspectives" in Journees systemes de reference spatio-temporels", 2004.

Appendix A. Contributors

This was an effort of the delay tolerant networking research group. The following dtnrg participants contributed significant technical material and/or inputs: Dr. Vinton Cerf of Google, Scott Burleigh, Adrian Hooke, and Leigh Torgerson of the Jet Propulsion Laboratory, Michael Demmer of the University of California at Berkeley, Robert Durst, Keith Scott, and Susan Symington of The MITRE Corporation, Kevin Fall of Intel Research, Stephen Farrell of Trinity College Dublin, Peter Lovell of SPARTA, Inc., Manikantan Ramadas of Ohio University (most of Section 4.1), and Howard Weiss of SPARTA, Inc. (text of Section 8) .

Appendix B. Comments

Please refer comments to dtn-interest@mailman.dtnrg.org. The Delay Tolerant Networking Research Group (DTNRG) web site is located at http://www.dtnrg.org.

Authors' Addresses

Keith L. Scott
The MITRE Corporation
7515 Colshire Drive
McLean, VA 21102
US

Phone: +1 703 983 6547
Fax: +1 703 983 7142
Email: kscott@mitre.org

Scott Burleigh
NASA Jet Propulsion Laboratory
4800 Oak Grove Dr.
Pasadena, CA 91109-8099
US

Phone: +1 818 393 3353
Fax: +1 818 354 1075
Email: Scott.Burleigh@jpl.nasa.gov

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).