

Delay-Tolerant Networking

Internet-Draft

Intended status: Informational

Expires: May 3, 2017

E. Birrane

Johns Hopkins Applied Physics Laboratory

October 30, 2016

Asynchronous Management Architecture
draft-birrane-dtn-ama-04

Abstract

This document describes the motivation, desirable properties, system model, roles/responsibilities, and component models associated with an asynchronous management architecture (AMA) suitable for providing application-level network management services in a challenged networking environment. Challenged networks are those that require fault protection, configuration, and performance reporting while unable to provide human-in-the-loop operations centers with synchronous feedback in the context of administrative sessions. In such a context, networks must exhibit behavior that is both determinable and autonomous while maintaining compatibility with existing network management protocols and operational concepts.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 3, 2017.

Copyright Notice

Copyright (c) 2016 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 Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Purpose	3
1.2. Scope	3
1.3. Requirements Language	4
1.4. ORganization	4
2. Terminology	5
3. Motivation	6
3.1. Challenged Networks	7
3.2. Current Management Approaches	8
3.3. Limitations of Current Approaches	8
4. Service Definitions	9
4.1. Configuration	9
4.2. Reporting	10
4.3. Autonomous Parameterized Control	10
4.4. Administration	11
5. Desirable Properties	11
5.1. Intelligent Push of Information	12
5.2. Minimize Message Size Not Node Processing	12
5.3. Absolute Data Identification	12
5.4. Custom Data Definition	13
5.5. Autonomous Operation	13
6. Roles and Responsibilities	13
6.1. Agent Responsibilities	13
6.2. Manager Responsibilities	15
7. System Model	15
7.1. Control and Data Flows	16
7.2. Control Flow by Role	16
7.2.1. Notation	17
7.2.2. Serialized Management	17
7.2.3. Multiplexed Management	18
7.2.4. Data Fusion	20
8. Logical Data Model	21
8.1. Data Decomposition	21
8.1.1. Groups	21
8.1.2. Levels	21
8.2. Data Model	22
8.2.1. EDDs, VARs, and Reporting	23
8.2.2. Controls and Macros	24
8.2.3. Rules	24

8.2.4. Operators and Literals	25
8.3. Application Data Model	25
9. IANA Considerations	26
10. Security Considerations	26
11. Informative References	26
Author's Address	27

1. Introduction

This document presents an Asynchronous Management Architecture (AMA) providing application-layer network management services over links where delivery delays prevent timely communications between a network operator and a managed device. These delays may be caused by long signal propagations or frequent link disruptions (such as described in [RFC4838]) or by non-environmental factors such as unavailability of network operators, administrative delays, or delays caused by quality-of-service prioritizations and service-level agreements.

1.1. Purpose

This document describes the motivation, rationale, desirable properties, and roles/responsibilities associated with an asynchronous management architecture (AMA) suitable for providing network management services in a challenged networking environment. These descriptions should be of sufficient specificity that an implementing Asynchronous Management Protocol (AMP) in conformance with this architecture will operate successfully in a challenged networking environment.

An AMA is necessary as the assumptions inherent to the architecture and design of synchronous management tools and techniques are not valid in challenged network scenarios. In these scenarios, synchronous approaches either patiently wait for periods of bi-directional connectivity or require the investment of significant time and resources to evolve a challenged network into a well-connected, low-latency network. In some cases such evolution is merely a costly way to over-resource a network. In other cases, such evolution is impossible given physical limitations imposed by signal propagation delays, power, transmission technologies, and other phenomena. Asynchronous management of asynchronous networks enables large-scale deployments, distributed technical capabilities, and reduced deployment and operations costs.

1.2. Scope

It is assumed that any challenged network where network management would be usefully applied supports basic services (where necessary) such as naming, addressing, integrity, confidentiality,

authentication, fragmentation, and traditional network/session layer functions. Therefore, these items are outside of the scope of the AMA and not covered in this document.

While likely that a challenged network will eventually interface with an unchallenged network, this document does not address the concept of network management compatibility with synchronous approaches. An AMP in conformance with this architecture should examine compatibility with existing approaches as part of supporting nodes acting as gateways between network types.

1.3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.4. ORganization

The remainder of this document is organizaed into seven sections that, together, describe an AMA suitable for enterprise management of asynchronous networks: terminology, motivation, service definitions, desirable properties, roles/responsibilities, system model, and logical component model. The description of each section is as follows.

- o Motivation - This section provides an overall motivation for this work as providing a novel and useful alternative to current network management approaches. Specifically, this section describes common network functions and how synchronous mechanisms fail to provide these functions in an asynchronous environment.
- o Service Definitions - This section defines asynchronous network management services in terms of terminology, scope, and impact.
- o Desirable Properties - This section identifies the properties to which an AMP should adhere to effectively implement service definitions in an asynchronous environment. These properties guide the subsequent definition of the system and logical models that comprise the AMA.
- o Roles and Responsibilities - This section identifies the roles of logical Actors in the AMA and their associated responsibilities. It provides the terminology and context for discussing how network management services interact.
- o System Model - This section describes data flows amongst various defined Actor roles. These flows capture how the AMA system works

to provide asynchronous network management services in accordance with defined desirable properties.

- o Logical Component Model - This section describes those logical functions that must exist in any instantiation of an AMP.

2. Terminology

This section identifies those terms critical to understanding the proper operation of the AMA. Whenever possible, these terms align in both word selection and meaning with their analogs from other management protocols.

- o Actor - A software service running on either managed or managing devices for the purpose of implementing management protocols between such devices. Actors may implement the "Manager" role, "Agent" role, or both.
- o Agent Role (or Agent) - The role associated with a managed device, responsible for reporting performance data, enforcing administrative policies, and accepting/performing actions. Agents exchange information with Managers operating either on the same device or on a remote managing device.
- o Asynchronous Management Protocol (AMP) - An application-layer protocol used to manage the Data, Controls, and other items necessary for configuration, monitoring, and administration of applications and protocols on a node in a challenged network.
- o Application Data Model (ADM) - The set of predefined data definitions, reports, literals, operations, and controls given to an Actor to manage a particular application or protocol. Actors support multiple ADMs, one for each application/protocol being managed.
- o Externally Defined Data (EDD) - Information made available to an Agent by a managed device, but not computed directly by the Agent. EDD definitions form the "lingua franca" for data within the AMA and are defined by ADMs.
- o Variable (VAR) - Information that is computed by an Agent, typically as a function of EDDs and/or other Variables. A VAR is a strongly-typed value. When a VAR is specified in an ADM, its type and default value are immutable. When a VAR is defined outside of an ADM, the type and default value may be changed. If an ADM wishes to define an item whose type and value are both immutable, that is no longer considered a Variable and should be represented as a Literal.

- o Controls (CTRLs) - Operations that may be undertaken by an Actor to change the behavior, configuration, or state of an application or protocol managed by an AMP. Similar to Externally Defined Data, Controls are defined solely in ADMs and their definition is immutable.
- o Literals (LIT) - Constants, enumerations, and other immutable definitions.
- o Macros - A named, ordered collection of Controls. When a Macro is defined in an ADM, that definition is immutable. When a Macro is defined outside of an ADM, that definition may be changed.
- o Manager - A role associated with a managing device responsible for configuring the behavior of, and receiving information from, Agents. Managers interact with one or more Agents located on the same device and/or on remote devices in the network.
- o Operator (OP) - The enumeration and specification of a mathematical function used to calculate computed data definitions and construct expressions to calculate state. Operators are specified in Application Data Models (ADMs) and their definition is immutable.
- o Report Entry (RPTE) - A named, typed, ordered collection of data values gathered by one or more Agents and provided to one or more Managers. Report entries populate report templates with values.
- o Report Template (RPTT) - An ordered collection of data identifiers. When defined in an ADM the report template definition is immutable. When defined outside of an ADM, the template definition may change.
- o Rule - A unit of autonomous specification that provides a stimulus-response relationship between time or state on an Agent and the Controls to be run as a result of that time or state.

3. Motivation

Challenged networks, to include networks challenged by administrative or policy delays, cannot guarantee capabilities required to enable synchronous management techniques. These capabilities include high-rate, highly-available data, round-trip data exchange, and operators "in-the-loop". The inability of current approaches to provide network management services in a challenged network motivates the need for a new network management architecture focused on asynchronous, open-loop, autonomous control of network components.

3.1. Challenged Networks

A growing variety of link-challenged networks support packetization to increase data communications reliability without otherwise guaranteeing a simultaneous end-to-end path. Examples of such networks include Mobile Ad-Hoc Networks (MANets), Vehicular Ad-Hoc Networks (VANets), Space-Terrestrial Internetworks (STINTs), and heterogeneous networking overlays. Links in such networks are often unavailable due to attenuations, propagation delays, occultation, and other limitations imposed by energy and mass considerations. Data communications in such networks rely on store-and-forward and other queueing strategies to wait for the connectivity necessary to usefully advance a packet along its route.

Similarly, there also exist well-resourced networks that incur high message delivery delays due to non-environmental limitations. For example, networks whose operations centers are understaffed or where data volume and management requirements exceed the real-time cognitive load of operators or the associated operations console software support. Also, networks where policy restricts user access to existing bandwidth creates situations functionally similar to link disruption and delay.

Independent of the reason, when a node experiences an inability to communicate it must rely on autonomous mechanisms to ensure its safe operation and ability to usefully re-join the network at a later time. In cases of sparsely-populated networks, there may never be a practical concept of "the connected network" as most nodes may be disconnected most of the time. In such environments, defining a network in terms of instantaneous connectivity becomes impractical or impossible.

Specifically, challenged networks exhibit the following properties that may violate assumptions built into current approaches to synchronous network management.

- o Links may be uni-directional.
- o Bi-directional links may have asymmetric data rates.
- o No end-to-end path is guaranteed to exist at any given time between any two nodes.
- o Round-trip communications between any two nodes within any given time window may be impossible.

3.2. Current Management Approaches

Network management tools in unchallenged networks provide mechanisms for communicating locally-collected data from Agents to Managers, typically using a "pull" mechanism where data must be explicitly requested by a Manager in order to be transmitted by an Agent.

A near ubiquitous method for management in unchallenged networks today is the Simple Network Management Protocol (SNMP) [RFC3416]. SNMP utilizes a request/response model to set and retrieve data values such as host identifiers, link utilizations, error rates, and counters between application software on Agents and Managers. Data may be directly sampled or consolidated into representative statistics. Additionally, SNMP supports a model for asynchronous notification messages, called traps, based on predefined triggering events. Thus, Managers can query Agents for status information, send new configurations, and be informed when specific events have occurred. Traps and queryable data are defined in one or more Managed Information Bases (MIBs) which define the information for a particular data standard, protocol, device, or application.

In challenged networks, the request/response method of data collection is neither efficient nor, at times, possible as it relies on sessions, round-trip latency, message retransmission, and ordered delivery. Adaptive modifications to SNMP to support challenged networks would alter the basic function of the protocol (data models, control flows, and syntax) so as to be functionally incompatible with existing SNMP installations.

The Network Configuration Protocol (NETCONF) provides device-level configuration capabilities [RFC6241] to replace vendor-specific command line interface (CLI) configuration software. The XML-based protocol provides a remote procedure call (RPC) syntax such that any exposed functionality on an Agent can be exercised via a software application interface. NETCONF places no specific functional requirements or constraints on the capabilities of the Agent, which makes it a very flexible tool for configuring a homogeneous network of devices. However, NETCONF does place specific constraints on any underlying transport protocol: namely, a long-lived, reliable, low-latency sequenced data delivery session. This is a fundamental requirement given the RPC-nature of the operating concept, and it is unsustainable in a challenged network.

3.3. Limitations of Current Approaches

Management approaches that rely on timely data exchange, such as those that rely on negotiated sessions or other synchronized acknowledgment, do not function in challenged network environments.

Familiar examples of TCP/IP based management via closed-loop, synchronous messaging does not work when network disruptions increase in frequency and severity. While no protocol delivers data in the absence of a networking link, protocols that eliminate or drastically reduce overhead and end-point coordination require smaller transmission windows and continue to function when confronted with scaling delays and disruptions in the network.

Just as the concept of a loosely-confederated set of nodes changes the definition of a network, it also changes the operational concept of what it means to manage a network. When a network stops being a single entity exhibiting a single behavior, "network management" becomes large-scale "node management". Individual nodes must share the burden of implementing desirable behavior without reliance on a single oracle of configuration or other coordinating function such as an operator-in-the-loop.

4. Service Definitions

This section identifies the services that must exist between Managers and Agents within an AMA. These services include configuration, reporting, parameterized control, and administration.

4.1. Configuration

Configuration services update local Agent information relating to managed applications and protocols. This information may be configured from ADMs, the specification of parameters associated with these models, and as defined by operators in the network.

New configurations received by a node must be validated to ensure that they do not conflict with other configurations at the node, or prevent the node from effectively working with other nodes in its region. With no guarantee of round-trip data exchange, Agents cannot rely on remote Managers to correct erroneous or stale configurations from harming the flow of data through a challenged network.

Examples of configuration service behavior include the following.

- o Creating a new datum as a function of other well-known data:
 $C = A + B$.
- o Creating a new report as a unique, ordered collection of known data:
 $RPT = \{A, B, C\}$.
- o Storing pre-defined, parameterized responses to potential future conditions:

```
IF (X > 3) THEN RUN CMD(PARM).
```

4.2. Reporting

Reporting services populate pre-defined Report Templates with values collected or computed by an Agent. The resultant Report Entries are sent to one or more Managers by the Agent. The term "reporting" is used in place of the term "monitoring", as monitoring implies a timeliness and regularity that cannot be guaranteed by a challenged network. Report Entries sent by an Agent provide best-effort information to receiving Managers.

Since a Manager is not actively "monitoring" an Agent, the Agent must make its own determination on when to send what Report Entries based on its own local time and state information. Agents should produce Report Entries of varying fidelity and with varying frequency based on thresholds and other information set as part of configuration services.

Examples of reporting service behavior include the following.

- o Generate Report Entry R1 every hour (time-based production).
- o Generate Report Entry R2 when $X > 3$ (state-based production).

4.3. Autonomous Parameterized Control

Controls represent a function that can be run by an Agent to affect its behavior or otherwise change its internal state. In this context, a Control may refer to a single function or an ordered set of functions run in sequence (e.g., a macro). The set of Controls understood by an Agent define the functions available to affect the behavior of applications and protocols managed by the Agent.

Since there is no guarantee that a Manager will be in contact with an Agent at any given time, the decisions of whether and when a Control should be run must be made locally and autonomously by the Agent. Two types of automation triggers are identified in the AMA: triggers based on the general state of the Agent and, more specifically, triggers based on an Agent's notion of time. As such, the autonomous execution of Controls can be viewed as a stimulus-response system, where the stimulus is the positive evaluation of a state or time based predicate and the response is the Control to be executed.

The autonomous nature of Control execution by an Agent implies that the full suite of information necessary to run a Control may not be known by a Manager in advance of running the Control on an Agent. To address this situation, Controls in the AMA MUST support a

parameterization mechanism so that required data can be provided at the time of execution on the Agent rather than at the time of definition/configuration by the Manager.

Autonomous, parameterized control provides a powerful mechanism for Managers to "manage" an Agent asynchronously during periods of no communication by pre-configuring responses to events that may be encountered by the Agent at a future time.

Examples of potential control service behavior include the following.

- o Updating local routing information based on instantaneous link analysis.
- o Managing storage on the device to enforce quotas.
- o Applying or modifying local security policy.

4.4. Administration

Administration services enforce the potentially complex mapping of configuration, reporting, and control services amongst Agents and Managers in the network. Fine-grained access control specifying which Managers may apply which services to which Agents may be necessary in networks dealing with multiple administrative entities or overlay networks crossing multiple administrative boundaries. Whitelists, blacklists, key-based infrastructures, or other schemes may be used for this purpose.

Examples of administration service behavior include the following.

- o Agent A1 only Sends reports for Protocol P1 to Manager M1.
- o Agent A2 only accepts a configurations for Application Y from Managers M2 and M3.
- o Agent A3 accepts services from any Manager providing the proper authentication token.

Note that the administrative enforcement of access control is different from security services provided by the networking stack carrying AMP messages.

5. Desirable Properties

This section describes those design properties that are desirable when defining an architecture that must operate across challenged links in a network. These properties ensure that network management

capabilities are retained even as delays and disruptions in the network scale. Ultimately, these properties are the driving design principles for the AMA.

5.1. Intelligent Push of Information

Pull management mechanisms require that a Manager send a query to an Agent and then wait for the response to that query. This practice implies a control-session between entities and increases the overall message traffic in the network. Challenged networks cannot guarantee timely roundtrip data-exchange and, in extreme cases, are comprised solely of uni-directional links. Therefore, pull mechanisms must be avoided in favor of push mechanisms.

Push mechanisms, in this context, refer to Agents making their own determinations relating to the information that should be sent to Managers. Such mechanisms do not require round-trip communications as Managers do not request each reporting instance; Managers need only request once, in advance, that information be produced in accordance with a pre-determined schedule or in response to a pre-defined state on the Agent. In this way information is "pushed" from Agents to Managers and the push is "intelligent" because it is based on some internal evaluation performed by the Agent.

5.2. Minimize Message Size Not Node Processing

Protocol designers must balance message size versus message processing time at sending and receiving nodes. Verbose representations of data simplify node processing whereas compact representations require additional activities to generate/parse the compacted message. There is no asynchronous management advantage to minimizing node processing time in a challenged network. However, there is a significant advantage to smaller message sizes in such networks. Compact messages require smaller periods of viable transmission for communication, incur less re-transmission cost, and consume less resources when persistently stored en-route in the network. AMPs should minimize PDUs whenever practical, to include packing and unpacking binary data, variable-length fields, and pre-configured data definitions.

5.3. Absolute Data Identification

Elements within the management system must be uniquely identifiable so that they can be individually manipulated. Identification schemes that are relative to system configuration make data exchange between Agents and Managers difficult as system configurations may change faster than nodes can communicate.

Consider the following SNMP technique for approximating an associative array lookup. A manager wishing to do an associative lookup for some key K1 will (1) query a list of array keys from the agent, (2) find the key that matched K1 and infer the index of K1 from the returned key list, and (3) query the discovered index on the agent to retrieve the desired data.

Ignoring the inefficiency of two pull requests, this mechanism fails when the Agent changes its key-index mapping between the first and second query. Rather than construting an artificial mapping from K1 to an index, an AMP must provide an absolute mechanism to lookup the value K1 without an abstraction between the Agent and Manager.

5.4. Custom Data Definition

Custom definition of new data from existing data (such as through data fusion, averaging, sampling, or other mechanisms) provides the ability to communicate desired information in as compact a form as possible. Specifically, an Agent should not be required to transmit a large data set for a Manager that only wishes to calculate a smaller, inferred data set. The Agent should calculate the smaller data set on its own and transmit that instead. Since the identification of custom data sets is likely to occur in the context of a specific network deployment, AMPs must provide a mechanism for their definition.

5.5. Autonomous Operation

AMA network functions must be achievable using only knowledge local to the Agent. Rather than directly controlling an Agent, a Manager configures the autonomy engine of the Agent to take its own action under the appropriate conditions in accordance with the Agent's notion of local state and time.

6. Roles and Responsibilities

By definition, Agents reside on managed devices and Managers reside on managing devices. This section describes how these roles participate in the network management functions outlined in the prior section.

6.1. Agent Responsibilities

Application Data Model (ADM) Support

Agents MUST collect all data, execute all Controls, populate all Report Templates and run operations required by each ADM which the Agent claims to support. Agents MUST report

supported ADMs so that Managers in a network understands what information is understood by what Agent.

Local Data Collection

Agents MUST collect from local firmware (or other on-board mechanisms) and report all Externally Defined Data defined in all ADMs for which they have been configured.

Autonomous Control

Agents MUST determine, without Manager intervention, whether a configured Control should be invoked. Agents MUST periodically evaluate the conditions associated with configured Controls and invoke those Controls based on local state. Agents MAY also invoke Controls on other devices for which they act as proxy.

User Data Definition

Agents MUST provide mechanisms for operators in the network to use configuration services to create customized Variables, Report Templates, Macros and other information in the context of a specific network or network use-case. Agents MUST allow for the creation, listing, and removal of such definitions in accordance with whatever security models are deployed within the particular network.

Where applicable, Agents MUST verify the validity of these definitions when they are configured and respond in a way consistent with the logging/error-handling policies of the Agent and the network.

Autonomous Reporting

Agents MUST determine, without real-time Manager intervention, whether and when to populate and transmit a given Report Entry targeted to one or more Managers in the network.

Consolidate Messages

Agents SHOULD produce as few messages as possible when sending information. For example, rather than sending multiple Report Entry messages to a Manager, an Agent SHOULD prefer to send a single message containing multiple Report Entries.

Regional Proxy

Agents MAY perform any of their responsibilities on behalf of other network nodes that, themselves, do not have an Agent. In such a configuration, the Agent acts as a proxy for these other network nodes.

6.2. Manager Responsibilities

Agent/ADM Mapping

Managers **MUST** understand what ADMs are supported by the various Agents with which they communicate. Managers should not attempt to request, invoke, or refer to ADM information for ADMs unsupported by an agent.

Data Collection

Managers **MUST** receive information from Agents by asynchronously configuring the production of data reports and then waiting for, and collecting, responses from Agents over time. Managers **MAY** try to detect conditions where Agent information has not been received within operationally relevant timespans and react in accordance with network policy.

Custom Definitions

Managers should provide the ability to define custom definitions. Any custom definitions **MUST** be transmitted to appropriate Agents and these definitions **MUST** be remembered to interpret the reporting of these custom values from Agents in the future.

Data Translation

Managers should provide some interface to other network management protocols, such as the SNMP. Managers **MAY** accomplish this by accumulating a repository of push-data from high-latency parts of the network from which data may be pulled by low-latency parts of the network.

Data Fusion

Managers **MAY** support the fusion of data from multiple Agents with the purpose of transmitting fused data results to other Managers within the network. Managers **MAY** receive fused reports from other Managers pursuant to appropriate security and administrative configurations.

7. System Model

This section describes the notional data flows and control flows that illustrate how Managers and Agents within an AMA cooperate to perform network management services.

7.1. Control and Data Flows

The AMA identifies three significant data flows: control flows from Managers to Agents, reports flows from Agents to Managers, and fusion reports from Managers to other Managers. These data flows are illustrated in Figure 1.

AMA Control and Data Flows

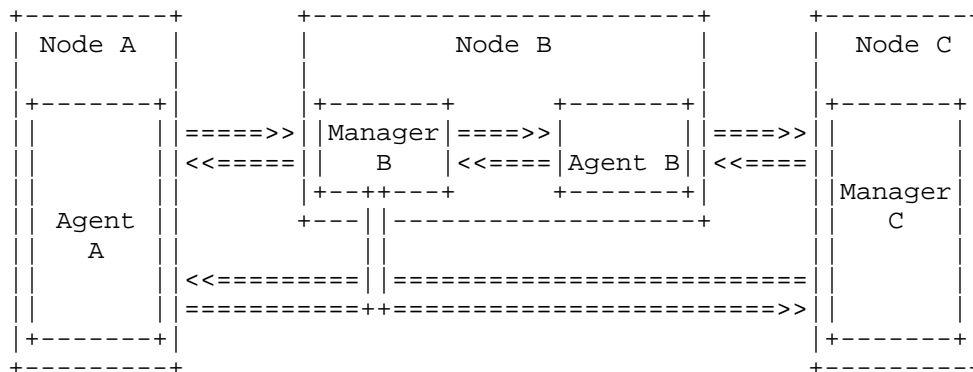


Figure 1

In this data flow, the Agent on node A receives Controls from Managers on nodes B and C, and replies with Report Entries back to these Managers. Similarly, the Agent on node B interacts with the local Manager on node B and the remote Manager on node C. Finally, the Manager on node B may fuse Report Entries received from Agents at nodes A and B and send these fused Report Entries back to the Manager on node C.

From this figure it is clear that there exist many-to-many relationships amongst Managers, amongst Agents, and between Agents and Managers. Note that Agents and Managers are roles, not necessarily differing software applications. Node A may represent a single software application fulfilling only the Agent role, whereas node B may have a single software application fulfilling both the Agent and Manager roles. The specifics of how these roles are realized is an implementation matter.

7.2. Control Flow by Role

This section describes three common configurations of Agents and Managers and the flow of messages between them. These configurations involve local and remote management and data fusion.

7.2.1. Notation

The notation outlined in Table 1 describes the types of control messages exchanged between Agents and Managers.

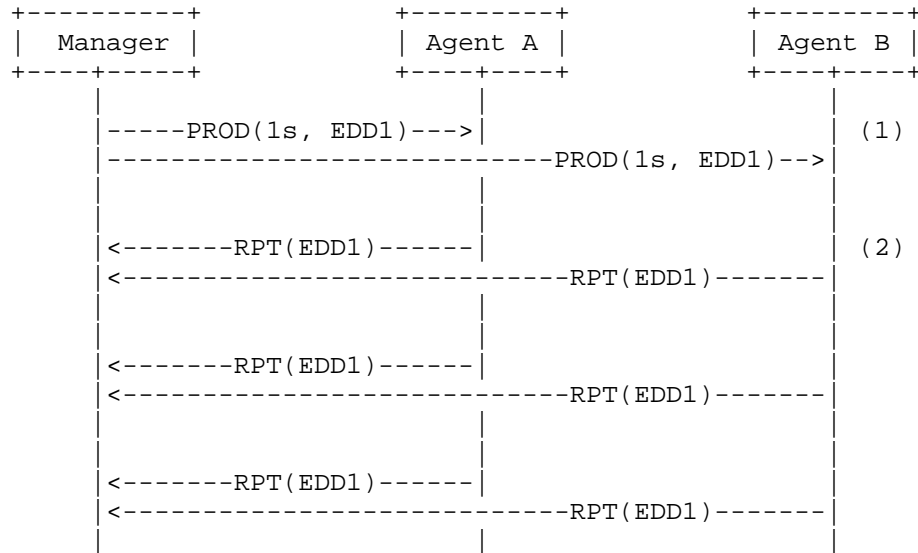
Term	Definition	Example
EDD#	EDD definition, from ADM.	EDD1
V#	Custom data definition.	V1 = EDD1 + V0.
DEF([ACL], ID,EXPR)	Define id from expression. Allow managers in access control list (ACL) to request this id.	DEF([*], V1, EDD1 + EDD2)
PROD(P,ID)	Produce ID according to predicate P. P may be a time period (1s) or an expression (EDD1 > 10).	PROD(1s, EDD1)
RPT(ID)	A report identified by ID.	RPT(EDD1)

Table 1: Terminology

7.2.2. Serialized Management

This is a nominal configuration of network management where a Manager interacts with a set of Agents. The control flows for this are outlined in Figure 2.

Serialized Management Control Flow



In a simple network, a Manager interacts with multiple Agents.

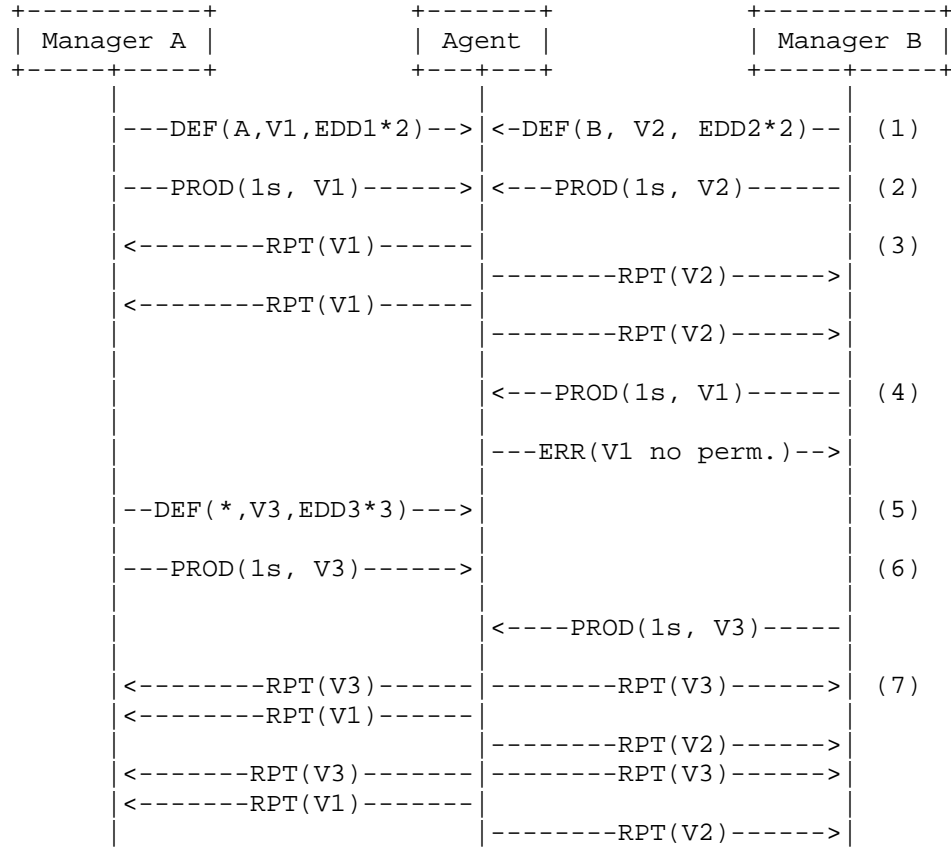
Figure 2

In this figure, the Manager configures Agents A and B to produce EDD1 every second in (1). At some point in the future, upon receiving and configuring this message, Agents A and B then build a Report Entry containing EDD1 and send those reports back to the Manager in (2).

7.2.3. Multiplexed Management

Networks spanning multiple administrative domains may require multiple Managers (for example, one per domain). When a Manager defines custom Report Templates/Variables to an Agent, that definition may be tagged with an access control list (ACL) to limit what other Managers will be privy to this information. Managers in such networks should synchronize with those other Managers granted access to their custom data definitions. When Agents generate messages, they MUST only send messages to Managers according to these ACLs, if present. The control flows in this scenario are outlined in Figure 3.

Multiplexed Management Control Flow



Complex networks require multiple Managers interfacing with Agents.

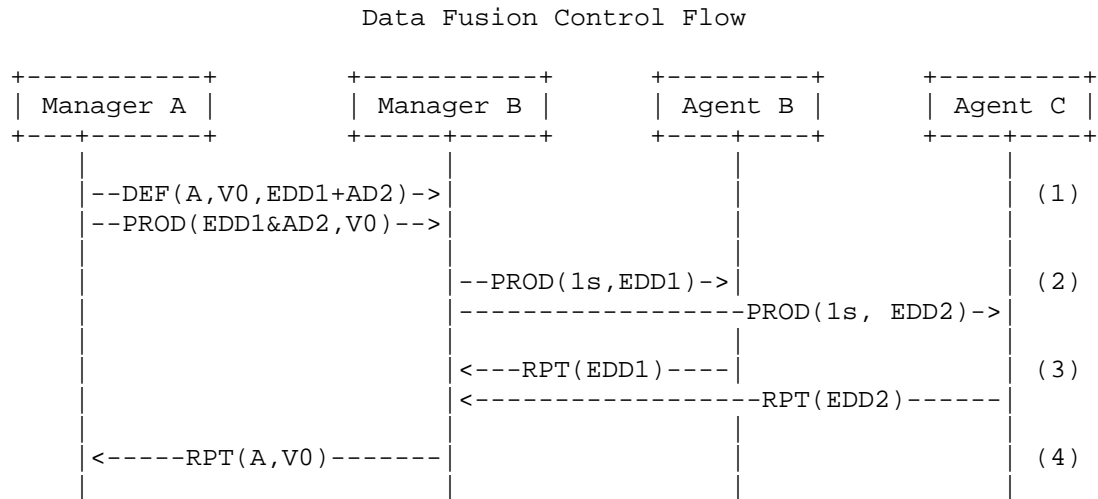
Figure 3

In more complex networks, any Manager may choose to define custom Report Templates and Variables, and Agents may need to accept such definitions from multiple Managers. Variable definitions may include an ACL that describes who may query and otherwise understand these definitions. In (1), Manager A defines V1 only for A while Manager B defines V2 only for B. Managers may, then, request the production of Report Entries containing these definitions, as shown in (2). Agents produce different data for different Managers in accordance with configured production rules, as shown in (3). If a Manager requests the production of a custom definition for which the Manager has no permissions, a response consistent with the configured logging policy on the Agent should be implemented, as shown in (4). Alternatively,

as shown in (5), a Manager may define custom data with no restrictions allowing all other Managers to request and use this definition. This allows all Managers to request the production of Report Entries containing this definition, shown in (6) and have all Managers receive this and other data going forward, as shown in (7).

7.2.4. Data Fusion

In some networks, Agents do not individually transmit their data to a Manager, preferring instead to fuse reporting data with local nodes prior to transmission. This approach reduces the number and size of messages in the network and reduces overall transmission energy expenditure. The AMA supports fusion of NM reports by co-locating Agents and Managers on nodes and offloading fusion activities to the Manager. This process is illustrated in Figure 4.



Data fusion occurs amongst Managers in the network.

Figure 4

In this example, Manager A requires the production of a Variable V0, from node B, as shown in (1). The Manager role understands what data is available from what agents in the subnetwork local to B, understanding that EDD1 is available locally and EDD2 is available remotely. Production messages are produced in (2) and data collected in (3). This allows the Manager at node B to fuse the collected Report Entries into V0 and return it in (4). While a trivial example, the mechanism of associating fusion with the Manager function rather than the Agent function scales with fusion complexity, though it is important to reiterate that Agent and

Manager designations are roles, not individual software components. There may be a single software application running on node B implementing both Manager B and Agent B roles.

8. Logical Data Model

This section identifies the different kinds of information present in an asynchronously-managed network and describes how this information should be communicated in the context of an ADM.

8.1. Data Decomposition

8.1.1. Groups

The AMA supports four basic groups of information: Data, Actions, Literals, and Operators:

Data Data values consist of information collected by an Agent and reported to Managers. This includes definitions from an ADM, derived data values as configured from Managers, and Report Entries which are collections of data elements.

Actions Actions are invoked on Agents and Managers to change behavior in response to some external event (such as local state changes or time). Actions include application-specific functions specified as part of an ADM and macros which are collections of these controls.

Literals Literals are constant numerical values that may be used in the evaluation of expressions and predicates.

Operators Operators are those mathematical functions that operate on series of Data and Literals, such as addition, subtraction, multiplication, and division.

8.1.2. Levels

The AMA defines three levels that describe the origins and multiplicity of data groups within the system. These classifications are atomic, computed, and collection.

Atomic

The Atomic level contains items computed or defined externally to the AMA and, thus, cannot be changed or otherwise decomposed by Actors within the AMA. These items are described in the context of an ADM and implemented in the context of firmware or software running on an Agent. The

identification of Atomic items MUST be globally unique and should be managed by a registration authority.

Computed

The Computed level contains items whose definition/value are specified/computed within the scope of an Actor in the AMA. Items at the computed level may be formally specified in an ADM (and therefore have definitions that are not subject to change) or may be defined dynamically on Agents by Managers and therefore have definitions that are subject to change in accordance with configuration services. In either case the definition of a Computed level item may reference other Computed level items and other Atomic level items if such inclusion does not result in a circular reference. When defined in the context of an ADM, a Computed level item MUST be globally unique and should be managed by a registration authority.

Collection

The Collection level contains items representing groups of other items, including other Collection level items. When a Collection level item definition references another Collection level item, circular references MUST be prevented. When defined in the context of an ADM, a Collection level item MUST be globally unique and should be managed by a registration authority.

8.2. Data Model

Each component of the AMA data model can be identified as a combination of group and level, as illustrated in Table 2. In this table, group/level combinations that are unsupported are listed as N/A. In this context, N/A indicates that the AMA does not require support for groups of data at a particular level for compliance.

	Data	Action	Literals	Operator
Atomic	Externally Defined Data	Control	Literal	Operator
Computed	Variable	Rule	N/A	N/A
Collection	Report Entry	Macro	N/A	N/A

Table 2

The eight elements of the AMA logical data model are described as follows.

8.2.1. EDDs, VARs, and Reporting

Fundamental to any performance reporting function is the ability to measure the state of the Agent. Measurement may be accomplished through direct sampling of hardware, query against in-situ data stores, or other mechanisms that provide the initial quantification of state.

EDDs serve as the "lingua franca" of the management system: the unit of information that cannot be otherwise created. As such, this information serves as the basis for any user-defined (Variable) values in the system.

AMPs MAY consider the concept of the confidence of the EDD as a function of time. For example, to understand at which point a measurement should be considered stale and need to be re-measured before acting on the associated data.

While EDDs provide the full, raw set of information available to Managers and Agents there is a performance optimization to pre-computed re-used combinations of these values. Computing new values as a function of measured values simplifies operator specifications and prevents Agent implementations from continuously re-calculating the same value each time it is used in a given time period.

For example, consider a sensor node which wishes to report a temperature averaged over the past 10 measurements. An Agent may either transmit all 10 measurements to a Manager, or calculate locally the average measurement and transmit the "fused" data. Clearly, the decision to reduce data volume is highly coupled to the nature of the science and the resources of the network. For this reason, the ability to define custom computations per deployment is necessary.

Periodically, or in accordance with local state changes, Agents must collect a series of measured values and computed values and communicate them back to Managers. This ordered collection of value information is noted in this architecture as a Report Entry which populates either a pre-defined or ad-hoc Report Template. In support of hierarchical definitions, Report Entries may, themselves, contain other Report Entries. It would be incumbent on an AMP implementation to guard against circular reference in Report Template definitions.

8.2.2. Controls and Macros

Just as traditional network management approaches provide well-known identifiers for values, the AMA provides well-known identifiers for Actions. Whereas several low-latency, high-availability approaches in networks can use approaches such as remote procedure calls (RPCs), challenged networks cannot provide a similar function - Managers cannot be in the processing loop of an Agent when the Agent is not in communication with the Manager.

Controls in a system are the combination of a well-known operation that can be taken by an Agent as well as any parameters that are necessary for the proper execution of that function. For specific applications or protocols, a control specification (as a series of opcodes) can be published such that any implementing AMP accepts these opcodes and understands that sending the opcodes to an Agent supporting the application or protocol will properly execute the associated function. Parameters to such functions are provided in real-time by either Managers requesting that a control be run, pre-configured, or auto-populated by the Agent in-situ.

Often, a series of controls must be executed in concert to achieve a particular function, especially when controls represent more primitive operations for a particular application/protocol. In such scenarios, an ordered collection of controls can be specified as a Macro. In support of the hierarchical build-up of functionality, Macros may, themselves, contain other Macros, through it would be incumbent on an AMP implementation to guard against excessive recursion or other resource-intensive nesting.

8.2.3. Rules

Stimulus-response autonomy systems provide a way to pre-configure responses to anticipated events. Such a mapping from responses to events is advantageous in a challenged network for a variety of reasons, as listed below.

- o Distributed Operation - The concept of pre-configuration allows the Agent to operate without regular contact with Managers in the system. Configuration opportunities will be sporadic in any challenged network making bootstrapping of the system difficult, but this is a fundamental problem in any network scenario and any autonomy approach.
- o Deterministic Behavior - Where the mapping of stimulus to response is stable, the behavior of the Agent to a variety of in-situ state also remains stable. This stable behavior is necessary in

critical operational systems where the actions of a platform must be well understood even in the absence of an operator in the loop.

- o Engine-Based Behavior - Several operational systems are unable to deploy "mobile code" based solutions due to network bandwidth, memory or processor loading, or security concerns. The benefit of engine-based approaches is that the configuration inputs to the engine can be flexible without incurring a set of problematic requirements or concerns.

The logical unit of stimulus-response autonomy proposed in the AMA is a Rule of the form:

IF stimulus THEN response

Where the set of such rules, when evaluated in some prioritized sequence, provides the full set of autonomous behavior for an Agent. Stimulus in such a system would either be a function of relative time, absolute time, or some mathematical expression comprising one or more values (measurement values or computed values).

Notably, in such a system, stimuli and responses from multiple applications and protocols may be combined to provide an expressive capability.

8.2.4. Operators and Literals

Computing values or evaluating expressions requires applying mathematical operations to data known to the management system.

Operators in the AMA represent enumerated mathematical operations applied to primitive and computed values in the AMA for the purpose of creating new values. Operations may be simple binary operations such as "A + B" or more complex functions such as sin(A) or avg(A,B,C,D).

Literals represent pre-configured constants in the AMA, such as well-known mathematical numbers (e.g., PI, E), or other useful data such as Epoch times. Literals also represent asserted Primitive Values used in expressions. For example, considering the expression (A = B + 10), A would be a Variable, B would be either a Variable or EDD, + would be an Operator, and 10 would be a Literal.

8.3. Application Data Model

Application data models (ADMs) specify the data associated with a particular application/protocol. The purpose of the ADM is to provide a published interface for the management of an application or protocol independent of the nuances of its software implementation. In this respect, the ADM is conceptually similar to the Managed

Information Base (MIB) used by SNMP, but contains additional information relating to command opcodes and more expressive syntax for automated behavior.

An ADM MUST define all well-known items necessary to manage the specific application or protocol. This includes the definitions of EDDs, Variables, Report Templates, Controls, Macros, Rules, Literals, and Operators.

9. IANA Considerations

At this time, this protocol has no fields registered by IANA.

10. Security Considerations

Security within an AMA MUST exist in two layers: transport layer security and access control.

Transport-layer security addresses the questions of authentication, integrity, and confidentiality associated with the transport of messages between and amongst Managers and Agents in the AMA. This security is applied before any particular Actor in the system receives data and, therefore, is outside of the scope of this document.

Finer grain application security is done via ACLs which are defined via configuration messages and implementation specific.

11. Informative References

[BIRrane1]

Birrane, E. and R. Cole, "Management of Disruption-Tolerant Networks: A Systems Engineering Approach", 2010.

[BIRrane2]

Birrane, E., Burleigh, S., and V. Cerf, "Defining Tolerance: Impacts of Delay and Disruption when Managing Challenged Networks", 2011.

[BIRrane3]

Birrane, E. and H. Kruse, "Delay-Tolerant Network Management: The Definition and Exchange of Infrastructure Information in High Delay Environments", 2011.

[I-D.irtf-dtnrg-dtnmp]

Birrane, E. and V. Ramachandran, "Delay Tolerant Network Management Protocol", draft-irtf-dtnrg-dtnmp-01 (work in progress), December 2014.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3416] Presuhn, R., Ed., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, DOI 10.17487/RFC3416, December 2002, <<http://www.rfc-editor.org/info/rfc3416>>.
- [RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, April 2007.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<http://www.rfc-editor.org/info/rfc6241>>.

Author's Address

Edward J. Birrane
Johns Hopkins Applied Physics Laboratory

Email: Edward.Birrane@jhuapl.edu

Delay-Tolerant Networking
Internet-Draft
Intended status: Informational
Expires: December 25, 2018

E. Birrane
Johns Hopkins Applied Physics Laboratory
June 23, 2018

Asynchronous Management Architecture
draft-birrane-dtn-ama-07

Abstract

This document describes an asynchronous management architecture (AMA) suitable for providing application-level network management services in a challenged networking environment. Challenged networks are those that require fault protection, configuration, and performance reporting while unable to provide humans-in-the-loop with synchronous feedback or otherwise preserve transport-layer sessions. In such a context, networks must exhibit behavior that is both determinable and autonomous while maintaining compatibility with existing network management protocols and operational concepts.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 25, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Scope	3
1.2. Requirements Language	4
1.3. Organization	4
2. Terminology	5
3. Motivation	6
3.1. Challenged Networks	7
3.2. Current Approaches and Their Limitations	8
4. Service Definitions	9
4.1. Configuration	9
4.2. Reporting	10
4.3. Autonomous Parameterized Procedure Calls	10
4.4. Administration	11
5. Desirable Properties	12
5.1. Intelligent Push of Information	12
5.2. Minimize Message Size Not Node Processing	12
5.3. Absolute Data Identification	13
5.4. Custom Data Definition	13
5.5. Autonomous Operation	13
6. Roles and Responsibilities	14
6.1. Agent Responsibilities	14
6.2. Manager Responsibilities	15
7. Logical Data Model	16
7.1. Data Representations: Constants, Externally Defined Data, and Variables	16
7.2. Data Collections: Reports and Tables	17
7.2.1. Report Templates and Reports	17
7.2.2. Table Templates and Tables	18
7.3. Command Execution: Controls and Macros	18
7.4. Autonomy: Time and State-Based Rules	19
7.4.1. State-Based Rule (SBR)	19
7.4.2. Time-Based Rule (TBR)	20
7.5. Calculations: Expressions, Literals, and Operators	20
8. System Model	21
8.1. Control and Data Flows	21
8.2. Control Flow by Role	22
8.2.1. Notation	22
8.2.2. Serialized Management	22
8.2.3. Multiplexed Management	23
8.2.4. Data Fusion	25
9. IANA Considerations	26

10. Security Considerations	26
11. Informative References	26
Author's Address	27

1. Introduction

The Asynchronous Management Architecture (AMA) provides application-layer network management services over links where delivery delays prevent timely communications between a network operator and a managed device. These delays may be caused by long signal propagations or frequent link disruptions (such as described in [RFC4838]) or by non-environmental factors such as unavailability of network operators, administrative delays, or delays caused by quality-of-service prioritizations and service-level agreements.

An AMA is necessary as the assumptions inherent to the architecture and design of synchronous management tools and techniques are not valid in challenged network scenarios. In these scenarios, synchronous approaches either patiently wait for periods of bi-directional connectivity or require the investment of significant time and resources to evolve a challenged network into a well-connected, low-latency network. In some cases such evolution is merely a costly way to over-resource a network. In other cases, such evolution is impossible given physical limitations imposed by signal propagation delays, power, transmission technologies, and other phenomena. Asynchronous management of asynchronous networks enables large-scale deployments, distributed technical capabilities, and reduced deployment and operations costs.

The rationale and motivation for asynchronous management is captured in [BIRrane1], [BIRrane2],[BIRrane3]. The properties and feasibility of such a system are taken from prototyping work done in accordance with [I-D.irtf-dtnrg-dtnmp].

1.1. Scope

This document describes the motivation, service definitions, desirable properties, roles/responsibilities, system model, and logical data model that form the AMA. These descriptions should be of sufficient specificity that implementations conformant to this architecture will operate successfully in a challenged networking environment.

This document is not a prescriptive standardization of a physical data model or protocol. Instead, it serves as informative guidance to authors of such models and protocols.

It is assumed that any challenged network where network management would be usefully applied supports basic services (where necessary) such as naming, addressing, integrity, confidentiality, authentication, fragmentation, and traditional network/session layer functions. Therefore, these items are outside of the scope of the AMA and not covered in this document.

While possible that a challenged network may interface with an unchallenged network, this document does not address the concept of network management compatibility with synchronous approaches.

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.3. Organization

The remainder of this document is organized into seven sections that, together, describe an AMA suitable for enterprise management of asynchronous networks: terminology, motivation, service definitions, desirable properties, roles/responsibilities, logical data model, and system model. The description of each section is as follows.

- o Terminology - This section identifies those terms critical to understanding the proper operation of the AMA. Whenever possible, these terms align in both word selection and meaning with their analogs from other management protocols.
- o Motivation - This section provides an overall motivation for this work as providing a novel and useful alternative to current network management approaches. Specifically, this section describes common network functions and how synchronous mechanisms fail to provide these functions in an asynchronous environment.
- o Service Definitions - This section defines asynchronous network management services in terms of terminology, scope, and impact.
- o Desirable Properties - This section identifies the properties to which an asynchronous management system should adhere to effectively implement service definitions in an asynchronous environment. These properties guide the subsequent definition of the system and logical models that comprise the AMA.
- o Roles and Responsibilities - This section identifies the roles in the AMA and their associated responsibilities. It provides the

terminology and context for discussing how network management services interact.

- o Logical Data Model - This section describes the kinds of data that should be represented in deployment asynchronous management system.
- o System Model - This section describes data flows amongst various defined Actor roles. These flows capture how the AMA system works to provide asynchronous network management services in accordance with defined desirable properties.

2. Terminology

- o Actor - A software service running on either managed or managing devices for the purpose of implementing management protocols between such devices. Actors may implement the "Manager" role, "Agent" role, or both.
- o Agent Role (or Agent) - The role associated with a managed device, responsible for reporting performance data, enforcing administrative policies, and accepting/performing actions. Agents exchange information with Managers operating either on the same device or on a remote managing device.
- o Externally Defined Data (EDD) - Information made available to an Agent by a managed device, but not computed directly by the Agent.
- o Variables (VARs) - Information that is computed by an Agent, typically as a function of EDD values and/or other Variables.
- o Constants (CONST) - A constant represents a typed, immutable value that is referred to by a semantic name. Constants are used in situations where substituting a name for a fixed value provides useful semantic information. For example, using the named constant PI rather than the literal value 3.14.
- o Controls (CTRLs) - Operations that may be undertaken by an Actor to change the behavior, configuration, or state of an application or protocol managed by an AMP.
- o Literals (LITs) - A literal represents a value without a semantic name. Literals are used in cases where adding a semantic name to a fixed value provides no useful semantic information. For example, the number 4 is a literal value.
- o Macros (MACs) - A named, ordered collection of Controls.

- o Manager - A role associated with a managing device responsible for configuring the behavior of, and receiving information from, Agents. Managers interact with one or more Agents located on the same device and/or on remote devices in the network.
- o Operator (OP) - The enumeration and specification of a mathematical function used to calculate variable values and construct expressions to evaluate Agent state.
- o Report (RPT) - A typed, ordered collection of data values gathered by one or more Agents and provided to one or more Managers. Reports only contain typed data values and the identity of the Report Template (RPTT) to which they conform.
- o Report Template (RPTT) - A named, typed, ordered collection of data types that represent the structure of a Report (RPT). This is the schema for a Report, generated by a Manager and communicated to one or more Agents.
- o Rule - A unit of autonomous specification that provides a stimulus-response relationship between time or state on an Agent and the Controls to be run as a result of that time or state.
- o State-Based Rule (SBR) - A state-based rule is any rule in which the rule stimulus is triggered by the calculable internal state of the Agent.
- o Table (TBL) - A typed collection of data values organized in a tabular way in which columns represent homogeneous types of data and rows represent unique sets of data values conforming to column types. Reports only contain typed data values and the identity of the Table Template (TBLT) to which they confirm.
- o Table Template (TBLT) - A named, typed, ordered collection of columns that comprise the structure for representing tabular data values. This template forms the structure of a Table (TBL).
- o Time-Based Rule (TBR) - A time-based rule is a specialization, and simplification, of a state-based rule in which the rule stimulus only considers relative time as it is known on the Agent.

3. Motivation

Challenged networks, to include networks challenged by administrative or policy delays, cannot guarantee capabilities required to enable synchronous management techniques. These capabilities include high-rate, highly-available data, round-trip data exchange, and operators "in-the-loop". The inability of current approaches to provide

network management services in a challenged network motivates the need for a new network management architecture focused on asynchronous, open-loop, autonomous control of network components.

3.1. Challenged Networks

A growing variety of link-challenged networks support packetization to increase data communications reliability without otherwise guaranteeing a simultaneous end-to-end path. Examples of such networks include Mobile Ad-Hoc Networks (MANets), Vehicular Ad-Hoc Networks (VANets), Space-Terrestrial Internetworks (STINTs), and heterogeneous networking overlays. Links in such networks are often unavailable due to attenuations, propagation delays, occultation, and other limitations imposed by energy and mass considerations. Data communications in such networks rely on store-and-forward and other queuing strategies to wait for the connectivity necessary to usefully advance a packet along its route.

Similarly, there also exist well-resourced networks that incur high message delivery delays due to hardware, software, or human limitations. Some examples of these networks are networks with understaffed operations centers and where data volume and management requirements exceed the real-time cognitive load of operators and/or their associated operations console software support. Also, networks that restrict user access to existing bandwidth due to policy create functionally similar situations to that of link disruption and delay.

Independent of the reason, when a node experiences an inability to communicate it must rely on autonomous mechanisms to ensure its safe operation and ability to usefully re-join the network at a later time. Additionally, nodes in a sparsely populated network may often be disconnected, making the concepts of "connected network" and "instantaneous connectivity" either impractical or impossible.

Specifically, challenged networks exhibit the following properties that may violate assumptions built into current approaches to synchronous network management.

- o Links may be uni-directional.
- o Bi-directional links may have asymmetric data rates.
- o No end-to-end path is guaranteed to exist at any given time between any two nodes.
- o Round-trip communications between any two nodes within any given time window may be impossible.

3.2. Current Approaches and Their Limitations

Network management tools in unchallenged networks provide mechanisms for communicating locally-collected data from Agents to Managers, typically using a "pull" mechanism where data must be explicitly requested by a Manager in order to be transmitted by an Agent.

Management approaches that rely on timely data exchange, such as those that rely on negotiated sessions or other synchronized acknowledgment, do not function in challenged network environments. Familiar examples of TCP/IP based management via closed-loop, synchronous messaging do not work when network disruptions increase in frequency and severity. While no protocol delivers data in the absence of a networking link, protocols that eliminate or drastically reduce overhead and end-point coordination require smaller transmission windows and continue to function when confronted with scaling delays and disruptions in the network.

A legacy method for management in unchallenged networks today is the Simple Network Management Protocol (SNMP) [RFC3416]. SNMP utilizes a request/response model to set and retrieve data values such as host identifiers, link utilizations, error rates, and counters between application software on Agents and Managers. Data may be directly sampled or consolidated into representative statistics. Additionally, SNMP supports a model for asynchronous notification messages, called traps, based on predefined triggering events. Thus, Managers can query Agents for status information, send new configurations, and be informed when specific events have occurred. Traps and queryable data are defined in one or more Managed Information Bases (MIBs) which define the information for a particular data standard, protocol, device, or application.

While there is a large installation base for SNMP there are several aspects of the protocol that make it inappropriate for use in a challenged networking environment. SNMP relies on sessions with low round-trip latency to support its "pull" model. The SNMP trap model provides some Agent-side processing, however because the processing has very low fidelity and traps are typically "fire and forget," the underlying transport protocol that supports reliable, in-order message delivery is required. Adaptive modifications to SNMP to support challenged networks would alter the basic function of the protocol (data models, control flows, and syntax) so as to be functionally incompatible with existing SNMP installations. Therefore, this approach is not suitable for an asynchronous network management system.

The Network Configuration Protocol (NETCONF) provides device-level configuration capabilities [RFC6241] to replace vendor-specific

command line interface configuration software. The XML-based protocol provides a remote procedure call (RPC) syntax such that any exposed functionality on an Agent can be exercised via a software application interface. NETCONF places no specific functional requirements or constraints on the capabilities of the Agent, which makes it a very flexible tool for configuring a homogeneous network of devices.

NETCONF places specific constraints on any underlying transport protocol: a long-lived, reliable, low-latency sequenced data delivery session. This is a fundamental requirement given the RPC-nature of the operating concept, and it is unsustainable in a challenged network. Aspects of the data modeling associated with NETCONF may apply to an asynchronous network management system, such that some modeling tools may be used, even if the network control plane cannot.

Just as the concept of a loosely-confederated set of nodes changes the definition of a network, it also changes the operational concept of what it means to manage a network. When a network stops being a single entity exhibiting a single behavior, "network management" becomes large-scale "node management". Individual nodes must share the burden of implementing desirable behavior without reliance on a single oracle of configuration or other coordinating function such as an operator-in-the-loop.

4. Service Definitions

This section identifies the services that must exist between Managers and Agents within an AMA. These services include configuration, reporting, parameterized control, and administration.

4.1. Configuration

Configuration services update Agent data associated with managed applications and protocols. Some configuration data might be defined in the context of an application or protocol, such that any network using that application or protocol would understand that data. Other configuration data may be defined tactically for use in a specific network deployment and not available to other networks even if they use the same applications or protocols.

New configurations received by an Agent must be validated to ensure that they do not conflict with other configurations or would otherwise prevent the Agent from effectively working with other Actors in its region. With no guarantee of round-trip data exchange, Agents cannot rely on remote Managers to correct erroneous or stale configurations from harming the flow of data through a challenged network.

Examples of configuration service behavior include the following.

- o Creating a new datum as a function of other well-known data:
C = A + B.
- o Creating a new report as a unique, ordered collection of known data:
RPT = {A, B, C}.
- o Storing predefined, parameterized responses to potential future conditions:
IF (X > 3) THEN RUN CMD(PARM).

4.2. Reporting

Reporting services populate report templates with values collected or computed by an Agent. The resultant reports are sent to one or more Managers by the Agent. The term "reporting" is used in place of the term "monitoring", as monitoring implies a timeliness and regularity that cannot be guaranteed by a challenged network. Reports sent by an Agent provide best-effort information to receiving Managers.

Since a Manager is not actively "monitoring" an Agent, the Agent must make its own determination on when to send what Reports based on its own local time and state information. Agents should produce Reports of varying fidelity and with varying frequency based on thresholds and other information set as part of configuration services.

Examples of reporting service behavior include the following.

- o Generate Report R1 every hour (time-based production).
- o Generate Report R2 when X > 3 (state-based production).

4.3. Autonomous Parameterized Procedure Calls

Similar to an RPC call, some mechanism MUST exist which allows a procedure to be run on an Agent in order to effect its behavior or otherwise change its internal state. Since there is no guarantee that a Manager will be in contact with an Agent at any given time, the decisions of whether and when a procedure should be run MUST be made locally and autonomously by the Agent. Two types of automation triggers are identified in the AMA: triggers based on the general state of the Agent and triggers based on an Agent's notion of time. As such, the autonomous execution of procedures can be viewed as a stimulus-response system, where the stimulus is the positive evaluation of a state or time based predicate and the response is the function to be executed.

The autonomous nature of procedure execution by an Agent implies that the full suite of information necessary to run a procedure may not be known by a Manager in advance. To address this situation, a parameterization mechanism MUST be available so that required data can be provided at the time of execution on the Agent rather than at the time of definition/configuration by the Manager.

Autonomous, parameterized procedure calls provide a powerful mechanism for Managers to "manage" an Agent asynchronously during periods of no communication by pre-configuring responses to events that may be encountered by the Agent at a future time.

Examples of potential behavior include the following.

- o Updating local routing information based on instantaneous link analysis.
- o Managing storage on the device to enforce quotas.
- o Applying or modifying local security policy.

4.4. Administration

Administration services enforce the potentially complex mapping of configuration, reporting, and control services amongst Agents and Managers in the network. Fine-grained access controls that specify which Managers may apply which services to which Agents may be necessary in networks that either deal with multiple administrative entities or overlay networks that cross administrative boundaries. Whitelists, blacklists, key-based infrastructures, or other schemes may be used for this purpose.

Examples of administration service behavior include the following.

- o Agent A1 only Sends reports for Protocol P1 to Manager M1.
- o Agent A2 only accepts a configurations for Application Y from Managers M2 and M3.
- o Agent A3 accepts services from any Manager providing the proper authentication token.

Note that the administrative enforcement of access control is different from security services provided by the networking stack carrying AMP messages.

5. Desirable Properties

This section describes those design properties that are desirable when defining an architecture that must operate across challenged links in a network. These properties ensure that network management capabilities are retained even as delays and disruptions in the network scale. Ultimately, these properties are the driving design principles for the AMA.

5.1. Intelligent Push of Information

Pull management mechanisms require that a Manager send a query to an Agent and then wait for the response to that query. This practice implies a control-session between entities and increases the overall message traffic in the network. Challenged networks cannot guarantee that the roundtrip data-exchange will occur in a timely fashion. In extreme cases, networks may be comprised of solely uni-directional links which drastically increases the amount of time needed for a roundtrip data exchange. Therefore, pull mechanisms must be avoided in favor of push mechanisms.

Push mechanisms, in this context, refer to the ability of Agents to make their own determinations in relation to the information that should be sent to Managers. Such mechanisms do not require round-trip communications as Managers do not request each reporting instance; Managers need only request once, in advance, that information be produced in accordance with a predetermined schedule or in response to a predefined state on the Agent. In this way information is "pushed" from Agents to Managers and the push is "intelligent" because it is based on some internal evaluation performed by the Agent.

5.2. Minimize Message Size Not Node Processing

Protocol designers must balance message size versus message processing time at sending and receiving nodes. Verbose representations of data simplify node processing whereas compact representations require additional activities to generate/parse the compacted message. There is no asynchronous management advantage to minimizing node processing time in a challenged network. However, there is a significant advantage to smaller message sizes in such networks. Compact messages require smaller periods of viable transmission for communication, incur less re-transmission cost, and consume less resources when persistently stored en-route in the network. AMPs should minimize PDUs whenever practical, to include packing and unpacking binary data, variable-length fields, and pre-configured data definitions.

5.3. Absolute Data Identification

Elements within the management system must be uniquely identifiable so that they can be individually manipulated. Identification schemes that are relative to system configuration make data exchange between Agents and Managers difficult as system configurations may change faster than nodes can communicate.

Consider the following common technique for approximating an associative array lookup. A manager wishing to do an associative lookup for some key K1 will (1) query a list of array keys from the agent, (2) find the key that matches K1 and infer the index of K1 from the returned key list, and (3) query the discovered index on the agent to retrieve the desired data.

Ignoring the inefficiency of two pull requests, this mechanism fails when the Agent changes its key-index mapping between the first and second query. Rather than constructing an artificial mapping from K1 to an index, an AMP must provide an absolute mechanism to lookup the value K1 without an abstraction between the Agent and Manager.

5.4. Custom Data Definition

Custom definition of new data from existing data (such as through data fusion, averaging, sampling, or other mechanisms) provides the ability to communicate desired information in as compact a form as possible. Specifically, an Agent should not be required to transmit a large data set for a Manager that only wishes to calculate a smaller, inferred data set. The Agent should calculate the smaller data set on its own and transmit that instead. Since the identification of custom data sets is likely to occur in the context of a specific network deployment, AMPs must provide a mechanism for their definition.

5.5. Autonomous Operation

AMA network functions must be achievable using only knowledge local to the Agent. Rather than directly controlling an Agent, a Manager configures an engine of the Agent to take its own action under the appropriate conditions in accordance with the Agent's notion of local state and time.

Such an engine may be used for simple automation of predefined tasks or to support semi-autonomous behavior in determining when to run tasks and how to configure or parameterize tasks when they are run. Wholly autonomous operations MAY be supported where required. Generally, autonomous operations should provide the following benefits.

- o Distributed Operation - The concept of pre-configuration allows the Agent to operate without regular contact with Managers in the system. The initial configuration (and periodic update) of the system remains difficult in a challenged network, but an initial synchronization on stimuli and responses drastically reduces needs for centralized operations.
- o Deterministic Behavior - Such behavior is necessary in critical operational systems where the actions of a platform must be well understood even in the absence of an operator in the loop. Depending on the types of stimuli and responses, these systems may be considered to be maintaining simple automation or semi-autonomous behavior. In either case, this preserves the ability of a frequently-out-of-contact Manager to predict the state of an Agent with more reliability than cases where Agents implement independent and fully autonomous systems.
- o Engine-Based Behavior - Several operational systems are unable to deploy "mobile code" based solutions due to network bandwidth, memory or processor loading, or security concerns. Engine-based approaches are preferred as they can be flexible without incurring a set of problematic requirements or concerns.

6. Roles and Responsibilities

By definition, Agents reside on managed devices and Managers reside on managing devices. This section describes how these roles participate in the network management functions outlined in the prior section.

6.1. Agent Responsibilities

Application Support

Agents MUST collect all data, execute all procedures, populate all reports and run operations required by each application which the Agent claims to manage. Agents MUST report supported applications so that Managers in a network understands what information is understood by what Agent.

Local Data Collection

Agents MUST collect from local firmware (or other on-board mechanisms) and report all data defined for the management of applications for which they have been configured.

Autonomous Control

Agents MUST determine, without Manager intervention, whether a procedure should be invoked. Agents MAY also invoke procedures on other devices for which they act as proxy.

User Data Definition

Agents MUST provide mechanisms for operators in the network to use configuration services to create customized data definitions in the context of a specific network or network use-case. Agents MUST allow for the creation, listing, and removal of such definitions in accordance with whatever security models are deployed within the particular network.

Where applicable, Agents MUST verify the validity of these definitions when they are configured and respond in a way consistent with the logging/error-handling policies of the Agent and the network.

Autonomous Reporting

Agents MUST determine, without real-time Manager intervention, whether and when to populate and transmit a given report targeted to one or more Managers in the network.

Consolidate Messages

Agents SHOULD produce as few messages as possible when sending information. For example, rather than sending multiple messages, each with one report to a Manager, an Agent SHOULD prefer to send a single message containing multiple reports.

Regional Proxy

Agents MAY perform any of their responsibilities on behalf of other network nodes that, themselves, do not have an Agent. In such a configuration, the Agent acts as a proxy for these other network nodes.

6.2. Manager Responsibilities

Agent Capabilities Mapping

Managers MUST understand what applications are managed by the various Agents with which they communicate. Managers should not attempt to request, invoke, or refer to application information for applications not managed by an Agent.

Data Collection

Managers MUST receive information from Agents by asynchronously configuring the production of reports and then waiting for, and collecting, responses from Agents over time. Managers MAY try to detect conditions where Agent information has not been received within operationally relevant time spans and react in accordance with network policy.

Custom Definitions

Managers should provide the ability to define custom data definitions. Any custom definitions MUST be transmitted to appropriate Agents and these definitions MUST be remembered to interpret the reporting of these custom values from Agents in the future.

Data Translation

Managers should provide some interface to other network management protocols. Managers MAY accomplish this by accumulating a repository of push-data from high-latency parts of the network from which data may be pulled by low-latency parts of the network.

Data Fusion

Managers MAY support the fusion of data from multiple Agents with the purpose of transmitting fused data results to other Managers within the network. Managers MAY receive fused reports from other Managers pursuant to appropriate security and administrative configurations.

7. Logical Data Model

The AMA logical data model captures the types of information that should be collected and exchanged to implement necessary roles and responsibilities. The data model presented in this section does not presuppose a specific mapping to a physical data model or encoding technique; it is included to provide a way to logically reason about the types of data that should be exchanged in an asynchronously managed network.

The elements of the AMA logical data model are described as follows.

7.1. Data Representations: Constants, Externally Defined Data, and Variables

There are three fundamental representations of data in the AMA: (1) data whose values do not change as a function of time or state, (2) data whose values change as determined by sampling/calculation external to the network management system, and (3) data whose values are calculated internal to the network management system.

Data whose values do not change as a function of time or state are defined as Constants (CONST). CONST values are strongly types, named values that cannot be modified once they have been defined.

Data that are sampled/calculated external to the network management system are defined as Externally Defined Data" (EDD). EDD values represent the most useful information in the management system as

they are provided by the applications or protocols being managed on the Agent. It is RECOMMENDED that EDD values be strongly typed to avoid issues with interpreting the data value. It is also RECOMMENDED that the timeliness/staleness of the data value be considered when using the data in the context of autonomous action on the Agent.

Data that is calculated internal to the network management system is defined as a Variable (VAR). VARs allow the creation of new data values for use in the network management system. New value definitions are useful for storing user-defined information, storing the results of complex calculations for easier re-use, and providing a mechanism for combining information from multiple external sources. It is RECOMMENDED that VARs be strongly typed to avoid issues with interpreting the data value. In cases where a VAR definition relies on other VAR definitions, mechanisms to prevent circular references MUST be included in any actual data model or implementation.

7.2. Data Collections: Reports and Tables

Individual data values may be exchanged amongst Agents and Managers in the AMA. However, data are typically most useful to a Manager when received as part of a set of information. Ordered collections of data values can be produced by Agents and sent to Managers as a way of efficiently communicating Agent status. Within the AMA, the structure of the ordered collection is treated separately from the values that populate such a structure.

The AMA provides two ways of defining collections of data: reports and tables. Reports are ordered sets of data values, whereas Tables are special types of reports whose entries have a regular, tabular structure.

7.2.1. Report Templates and Reports

The typed, ordered structure of a data collection is defined as a Report Template (RPTT). A particular set of data values provided in compliance with such a template is called a Report (RPT).

Separating the structure and content of a report reduces the overall size of RPTs in cases where reporting structures are well known and unchanging. RPTTs can be synchronized between an Agent and a Manager so that RPTs themselves do not incur the overhead of carrying self-describing data. RPTTs may include EDD values, VARs, and also other RPTTs. In cases where a RPTT includes another RPTTs, mechanisms to prevent circular references MUST be included in any actual data model or implementation.

Protocols and applications managed in the AMA may define common RPTTs. Additionally, users within a network may define their own RPTTs that are useful in the context of a particular deployment.

7.2.2. Table Templates and Tables

Tables optimize the communication of multiple sets of data in situations where each data set has the same syntactical structure and with the same semantic meaning. Unlike reports, the regularity of tabular data representations allow for the addition of new rows without changing the structure of the table. Attempting to add a new data set at the end of a report would require alterations to the report template.

The typed, ordered structure of a table is defined as a Table Template (TBLT). A particular instance of values populating the table template is called a Table (TBL).

TBLTs describes the "columns" that define the table schema. A TBL represents the instance of a specific TBLT that holds actual data values. These data values represent the "rows" of the table.

The prescriptive nature of the TBLT allows for the possibility of advanced filtering which may reduce traffic between Agents and Managers. However, the unique structure of each TBLT may make them difficult or impossible to change dynamically in a network.

7.3. Command Execution: Controls and Macros

Low-latency, high-availability approaches to network management use mechanisms such as (or similar to) RPCs to cause some action to be performed on an Agent. The AMA enables similar capabilities without requiring that the Manager be in the processing loop of the Agent. Command execution in the AMA happens through the use of controls and macros.

A Control (CTRL) represents a parameterized, predefined procedure that can be run on an Agent. CTRLs do not have a return code as there is not the same concept of sequential execution in an asynchronous model. Parameters can be provided when running a command from a Manager, pre-configured as part of an autonomy response on the Agent, or auto-generated as needed on the Agent. The success or failure of a control MAY be inferred by reports generated for that purpose.

NOTE: The AMA term control is derived in part from the concept of Command and Control (C2) where control implies the operational instructions that must be undertaken to implement (or maintain) a

commanded objective. An asynchronous management function controls an Agent to allow it to fulfill its commanded purpose in a variety of operational scenarios. For example, attempting to maintain a safe internal thermal environment for a spacecraft is considered "thermal control" (not "thermal commanding") even though thermal control involves "commanding" heaters, louvers, radiators, and other temperature-affecting components.

Often, a series of controls must be executed in concert to achieve a particular outcome. A Macro (MACRO) represents an ordered collection of controls (or other macros). In cases where a MACRO includes another MACRO, mechanisms to prevent circular references and maximum nesting levels MUST be included in any actual data model or implementation.

7.4. Autonomy: Time and State-Based Rules

The AMA data model contains EDDs and VARs that capture the state of applications on an Agent. The model also contains controls and macros to perform actions on an Agent. A mechanism is needed to relate these two capabilities: to perform an action on the Agent in response to the state of the Agent. This mechanism in the AMA is the "rule" and can be key activated based on Agent state (state-based rule) or based on the Agent's notion of relative time (time-based rule).

7.4.1. State-Based Rule (SBR)

State-Based Rules (SBRs) perform actions based on the Agent's internal state, as identified by EDD and VAR values. An SBR represents a stimulus-response pairing in the following form:

IF predicate THEN response

The predicate is a logical expression that evaluates to true if the rule stimulus is present and evaluates to false otherwise. The response may be any control or macro known to the Agent.

An example of an SBR could be to turn off a heater if some internal temperature is greater than a threshold:

IF (current_temp > maximum_temp) THEN turn_heater_off

Rules should be allowed to construct their stimuli from the full set of EDD values and VARs available to the network management system. Similarly, macro responses should be allowed to include controls from all applications known by the Agent. This enables an expressive capability to have multiple applications monitored and managed by the Agent.

7.4.2. Time-Based Rule (TBR)

Time-Based Rules (TBR) perform actions based on the Agent's notion of the passage of time. A possible TBR construct would be to perform some action at 1Hz on the Agent.

A TBR is a specialization of an SBR as the Agent's notion of time is a type of Agent state. For example, a TBR to perform an action every 24 hours could be expressed using some type of predicate of the form:

```
((current_time - base_time) % 24_hours) == 0)
```

However, time-based events are popular enough that special semantics for expressing them would likely significantly reduce the computations necessary to represent time functions in a SBR.

7.5. Calculations: Expressions, Literals, and Operators

Actions such as computing a VAR value or describing a rule predicate require some mechanism for calculating the value of mathematical expressions. In addition the the aforementioned AMA logical data objects, Literals, Operators, and Expressions are used to perform these calculations.

A Literal (LIT) represents a strongly typed datum whose identity is equivalent to its value. An example of a LIT value is "4" - it's identifier (4) is the same as its value (4). Literals differ from constants in that constants have an identifier separate from their value. For example, the constant PI may refer to a value of 3.14. However the literal 3.14159 always refers to the value 3.14159.

An Operator (OP) represents a mathematical operation in an expression. OPs should support multiple operands based on the operation supported. A common set of OPs SHOULD be defined for any Agent and systems MAY choose to allow individual applications to define new OPs to assist in the generation of new VAR values and predicates for managing that application. OPs may be simple binary operations such as "A + B" or more complex functions such as sin(A) or avg(A,B,C,D). Additionally, OPs may be typed. For example,

addition of integers may be defined separately from addition of real numbers.

An Expression (EXPR) is a combination of operators and operands used to construct a numerical value from a series of other elements of the AMA logical model. Operands include any AMA logical data model object that can be interpreted as a value, such as EDD, VAR, CONST, and LIT values. Operators perform some function on operands to generate new values.

8. System Model

This section describes the notional data flows and control flows that illustrate how Managers and Agents within an AMA cooperate to perform network management services.

8.1. Control and Data Flows

The AMA identifies three significant data flows: control flows from Managers to Agents, reports flows from Agents to Managers, and fusion reports from Managers to other Managers. These data flows are illustrated in Figure 1.

AMA Control and Data Flows

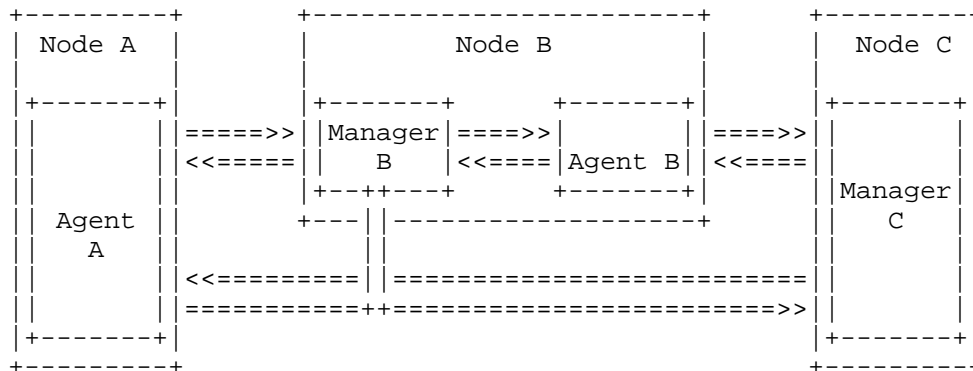


Figure 1

In this data flow, the Agent on node A receives Controls from Managers on nodes B and C, and replies with Report Entries back to these Managers. Similarly, the Agent on node B interacts with the local Manager on node B and the remote Manager on node C. Finally, the Manager on node B may fuse Report Entries received from Agents at nodes A and B and send these fused Report Entries back to the Manager on node C.

From this figure it is clear that there exist many-to-many relationships amongst Managers, amongst Agents, and between Agents and Managers. Note that Agents and Managers are roles, not necessarily differing software applications. Node A may represent a single software application fulfilling only the Agent role, whereas node B may have a single software application fulfilling both the Agent and Manager roles. The specifics of how these roles are realized is an implementation matter.

8.2. Control Flow by Role

This section describes three common configurations of Agents and Managers and the flow of messages between them. These configurations involve local and remote management and data fusion.

8.2.1. Notation

The notation outlined in Table 1 describes the types of control messages exchanged between Agents and Managers.

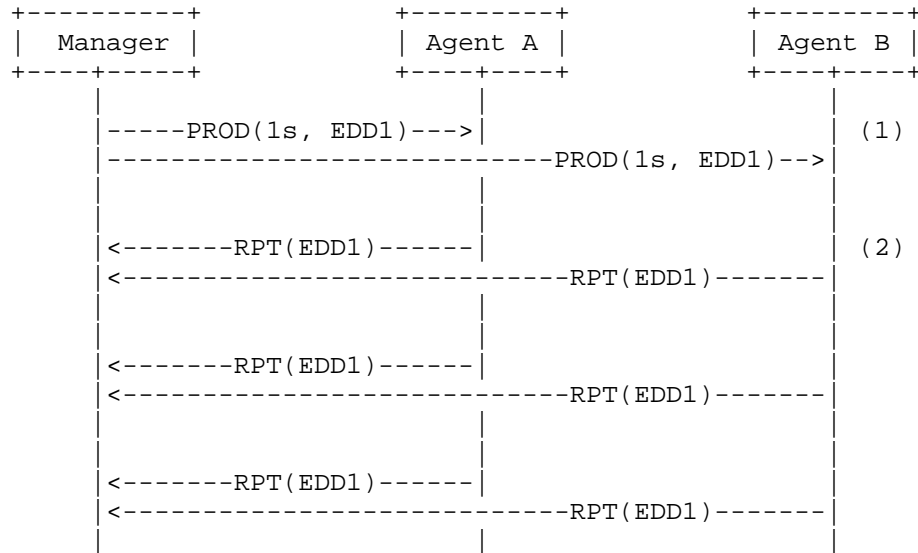
Term	Definition	Example
EDD#	EDD definition.	EDD1
V#	Variable definition.	V1 = EDD1 + V0.
DEF([ACL], ID,EXPR)	Define id from expression. Allow managers in access control list (ACL) to request this id.	DEF([*], V1, EDD1 + EDD2)
PROD(P,ID)	Produce ID according to predicate P. P may be a time period (1s) or an expression (EDD1 > 10).	PROD(1s, EDD1)
RPT(ID)	A report identified by ID.	RPT(EDD1)

Table 1: Terminology

8.2.2. Serialized Management

This is a nominal configuration of network management where a Manager interacts with a set of Agents. The control flows for this are outlined in Figure 2.

Serialized Management Control Flow



In a simple network, a Manager interacts with multiple Agents.

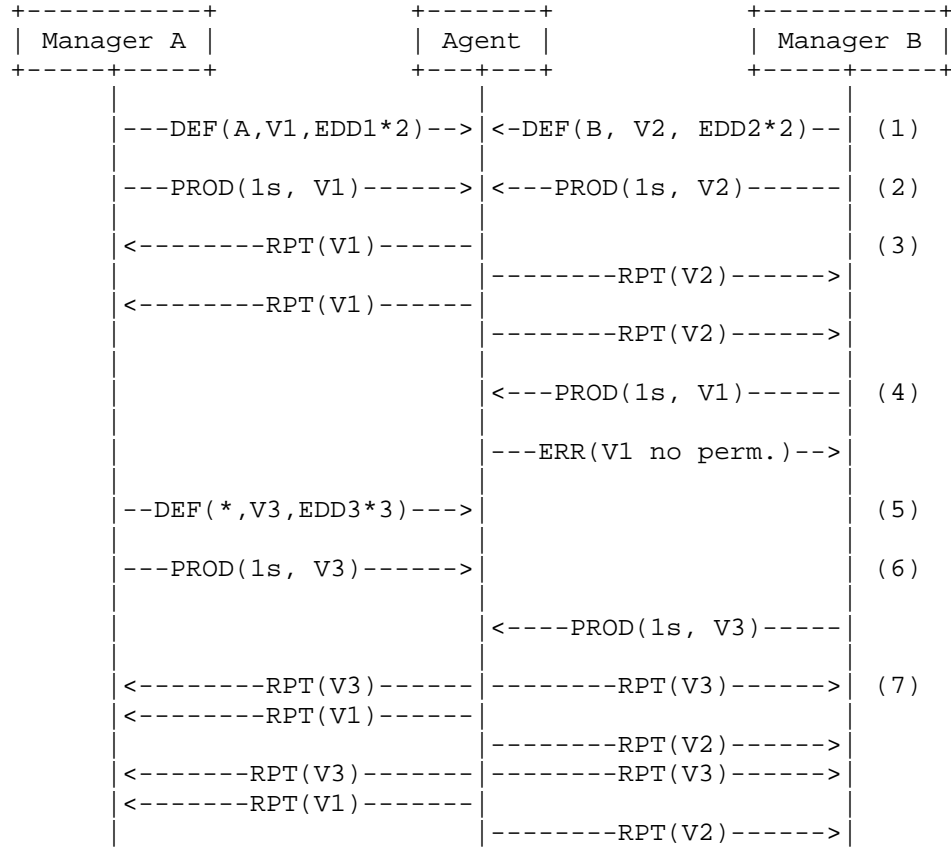
Figure 2

In this figure, the Manager configures Agents A and B to produce EDD1 every second in (1). At some point in the future, upon receiving and configuring this message, Agents A and B then build a Report Entry containing EDD1 and send those reports back to the Manager in (2).

8.2.3. Multiplexed Management

Networks spanning multiple administrative domains may require multiple Managers (for example, one per domain). When a Manager defines custom Reports/Variables to an Agent, that definition may be tagged with an Access Control List (ACL) to limit what other Managers will be privy to this information. Managers in such networks should synchronize with those other Managers granted access to their custom data definitions. When Agents generate messages, they MUST only send messages to Managers according to these ACLs, if present. The control flows in this scenario are outlined in Figure 3.

Multiplexed Management Control Flow



Complex networks require multiple Managers interfacing with Agents.

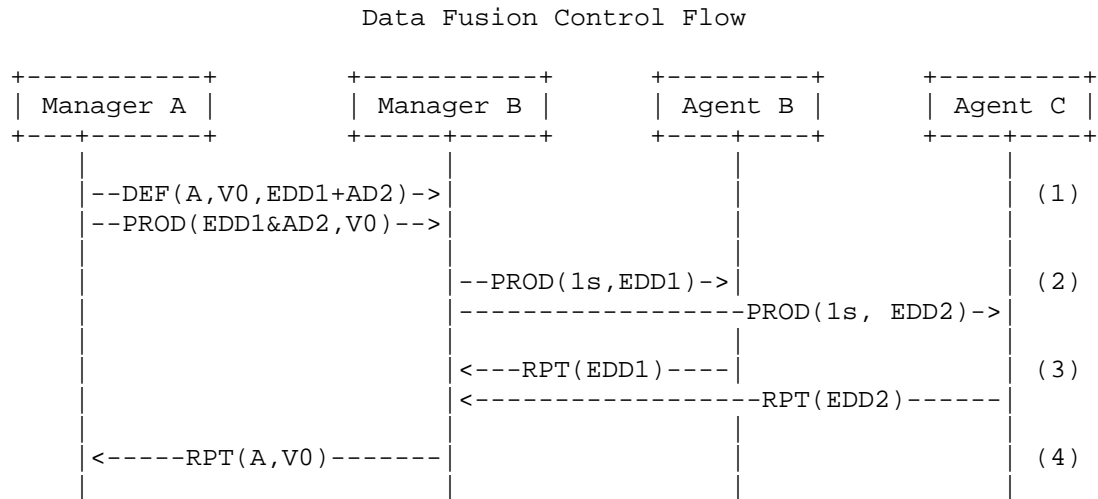
Figure 3

In more complex networks, any Manager may choose to define custom Reports and Variables, and Agents may need to accept such definitions from multiple Managers. Variable definitions may include an ACL that describes who may query and otherwise understand these definitions. In (1), Manager A defines V1 only for A while Manager B defines V2 only for B. Managers may, then, request the production of Report Entries containing these definitions, as shown in (2). Agents produce different data for different Managers in accordance with configured production rules, as shown in (3). If a Manager requests the production of a custom definition for which the Manager has no permissions, a response consistent with the configured logging policy on the Agent should be implemented, as shown in (4). Alternatively,

as shown in (5), a Manager may define custom data with no restrictions allowing all other Managers to request and use this definition. This allows all Managers to request the production of Report Entries containing this definition, shown in (6) and have all Managers receive this and other data going forward, as shown in (7).

8.2.4. Data Fusion

In some networks, Agents do not individually transmit their data to a Manager, preferring instead to fuse reporting data with local nodes prior to transmission. This approach reduces the number and size of messages in the network and reduces overall transmission energy expenditure. The AMA supports fusion of NM reports by co-locating Agents and Managers on nodes and offloading fusion activities to the Manager. This process is illustrated in Figure 4.



Data fusion occurs amongst Managers in the network.

Figure 4

In this example, Manager A requires the production of a Variable V0, from node B, as shown in (1). The Manager role understands what data is available from what agents in the subnetwork local to B, understanding that EDD1 is available locally and EDD2 is available remotely. Production messages are produced in (2) and data collected in (3). This allows the Manager at node B to fuse the collected Report Entries into V0 and return it in (4). While a trivial example, the mechanism of associating fusion with the Manager function rather than the Agent function scales with fusion complexity, though it is important to reiterate that Agent and

Manager designations are roles, not individual software components. There may be a single software application running on node B implementing both Manager B and Agent B roles.

9. IANA Considerations

This protocol has no fields registered by IANA.

10. Security Considerations

Security within an AMA MUST exist in two layers: transport layer security and access control.

Transport-layer security addresses the questions of authentication, integrity, and confidentiality associated with the transport of messages between and amongst Managers and Agents in the AMA. This security is applied before any particular Actor in the system receives data and, therefore, is outside of the scope of this document.

Finer grain application security is done via ACLs which are defined via configuration messages and implementation specific.

11. Informative References

[BIRrane1]

Birrane, E. and R. Cole, "Management of Disruption-Tolerant Networks: A Systems Engineering Approach", 2010.

[BIRrane2]

Birrane, E., Burleigh, S., and V. Cerf, "Defining Tolerance: Impacts of Delay and Disruption when Managing Challenged Networks", 2011.

[BIRrane3]

Birrane, E. and H. Kruse, "Delay-Tolerant Network Management: The Definition and Exchange of Infrastructure Information in High Delay Environments", 2011.

[I-D.irtf-dtnrg-dtnmp]

Birrane, E. and V. Ramachandran, "Delay Tolerant Network Management Protocol", draft-irtf-dtnrg-dtnmp-01 (work in progress), December 2014.

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC3416] Presuhn, R., Ed., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, DOI 10.17487/RFC3416, December 2002, <<https://www.rfc-editor.org/info/rfc3416>>.
- [RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, April 2007.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.

Author's Address

Edward J. Birrane
Johns Hopkins Applied Physics Laboratory

Email: Edward.Birrane@jhuapl.edu

Delay-Tolerant Networking Working Group
Internet Draft
Intended status: Standards Track
Expires: May 2017

S. Burleigh
JPL, Calif. Inst. Of Technology
K. Fall
Carnegie Mellon University / SEI
E. Birrane
APL, Johns Hopkins University
October 29, 2016

Bundle Protocol
draft-ietf-dtn-bpbis-06.txt

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), 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/lid-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 2, 2017.

Copyright Notice

Copyright (c) 2016 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 Simplified BSD License.

Abstract

This Internet Draft presents a specification for Bundle Protocol, adapted from the experimental Bundle Protocol specification developed by the Delay-Tolerant Networking Research group of the Internet Research Task Force and documented in RFC 5050.

Table of Contents

1. Introduction.....	3
2. Conventions used in this document.....	5
3. Service Description.....	5
3.1. Definitions.....	5
3.2. Discussion of BP concepts.....	9
3.3. Services Offered by Bundle Protocol Agents.....	14
4. Bundle Format.....	14
4.1. BP Fundamental Data Structures.....	15
4.1.1. CRC Type.....	15
4.1.2. CRC.....	15
4.1.3. Bundle Processing Control Flags.....	15
4.1.4. Block Processing Control Flags.....	17
4.1.5. Identifiers.....	18
4.1.5.1. Endpoint ID.....	18
4.1.5.2. Node ID.....	19
4.1.6. DTN Time.....	19
4.1.7. Creation Timestamp.....	19
4.1.8. Block-type-specific Data.....	19
4.2. Bundle Representation.....	20
4.2.1. Bundle.....	20
4.2.2. Primary Bundle Block.....	20
4.2.3. Canonical Bundle Block Format.....	22
4.3. Extension Blocks.....	23
4.3.1. Current Custodian.....	24
4.3.2. Previous Node.....	24
4.3.3. Bundle Age.....	24
4.3.4. Hop Count.....	25
5. Bundle Processing.....	25
5.1. Generation of Administrative Records.....	25
5.2. Bundle Transmission.....	26
5.3. Bundle Dispatching.....	27
5.4. Bundle Forwarding.....	27
5.4.1. Forwarding Contraindicated.....	29
5.4.2. Forwarding Failed.....	29

5.5. Bundle Expiration.....	30
5.6. Bundle Reception.....	30
5.7. Local Bundle Delivery.....	31
5.8. Bundle Fragmentation.....	32
5.9. Application Data Unit Reassembly.....	33
5.10. Custody Transfer.....	34
5.10.1. Custody Acceptance.....	34
5.10.2. Custody Release.....	35
5.11. Custody Transfer Success.....	35
5.12. Custody Transfer Failure.....	35
5.13. Custody Transfer Deferral.....	36
5.14. Bundle Deletion.....	36
5.15. Discarding a Bundle.....	37
5.16. Canceling a Transmission.....	37
6. Administrative Record Processing.....	37
6.1. Administrative Records.....	37
6.1.1. Bundle Status Reports.....	38
6.1.2. Custody Signals.....	40
6.2. Generation of Administrative Records.....	43
6.3. Reception of Custody Signals.....	44
7. Services Required of the Convergence Layer.....	44
7.1. The Convergence Layer.....	44
7.2. Summary of Convergence Layer Services.....	44
8. Security Considerations.....	45
9. IANA Considerations.....	46
10. References.....	46
10.1. Normative References.....	46
10.2. Informative References.....	47
11. Acknowledgments.....	47
12. Significant Changes from RFC 5050.....	48
Appendix A. For More Information.....	49
Appendix B. CDDL expression.....	50

1. Introduction

Since the publication of the Bundle Protocol Specification (Experimental RFC 5050[RFC5050]) in 2007, the Delay-Tolerant Networking Bundle Protocol has been implemented in multiple programming languages and deployed to a wide variety of computing platforms for a wide range of successful exercises. This implementation and deployment experience has demonstrated the general utility of the protocol for challenged network operations.

It has also, as expected, identified opportunities for making the protocol simpler, more capable, and easier to use. The present document, standardizing the Bundle Protocol (BP), is adapted from RFC 5050 in that context.

This document describes version 7 of BP.

Delay Tolerant Networking is a network 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 may be viewed as sitting at the application layer of some number of constituent networks, forming a store-carry-forward overlay network. Key capabilities of BP include:

- . Custodial forwarding
- . Ability to cope with intermittent connectivity, including cases where the sender and receiver are not concurrently present in the network
- . Ability to take advantage of scheduled, predicted, and opportunistic connectivity, whether bidirectional or unidirectional, in addition to continuous connectivity
- . Late binding of overlay network endpoint identifiers to underlying constituent network 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.

BP's location within the standard protocol stack is as shown in Figure 1. BP uses underlying "native" transport and/or network protocols for communications within a given constituent network.

The interface between the bundle protocol and a specific underlying protocol is termed a "convergence layer adapter".

Figure 1 shows three distinct transport and network protocols (denoted T1/N1, T2/N2, and T3/N3).

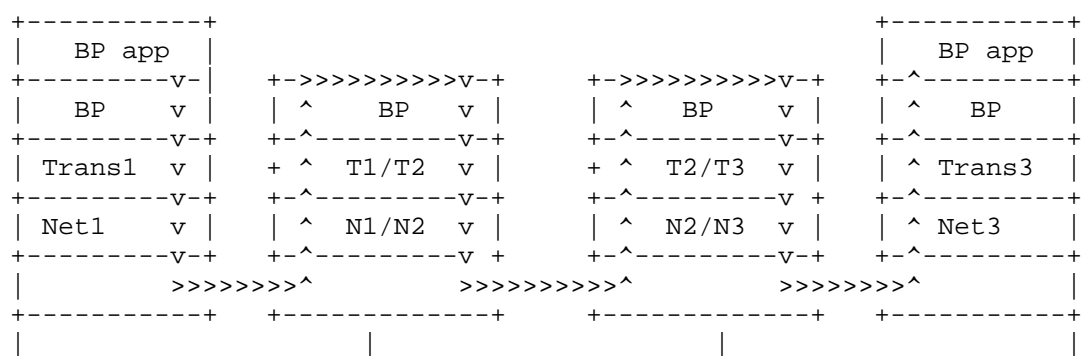




Figure 1: The Bundle Protocol in the Protocol Stack Model

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 internets. (However, the document does discuss the services that must be provided by each adapter at the convergence layer.)
- . The bundle route computation algorithm.
- . Mechanisms for populating the routing or forwarding information bases of bundle nodes.
- . The mechanisms for securing bundles en route.
- . The mechanisms for managing bundle nodes.

2. Conventions used in this document

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 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

3. Service Description

3.1. Definitions

Bundle - A bundle is a protocol data unit of BP, so named because negotiation of the parameters of a data exchange may be impractical in a delay-tolerant network: it is often better practice to "bundle" with a unit of data all metadata that might be needed in order to make the data immediately usable when delivered to applications. Each bundle comprises a sequence of two or more "blocks" of protocol data, which serve various purposes.

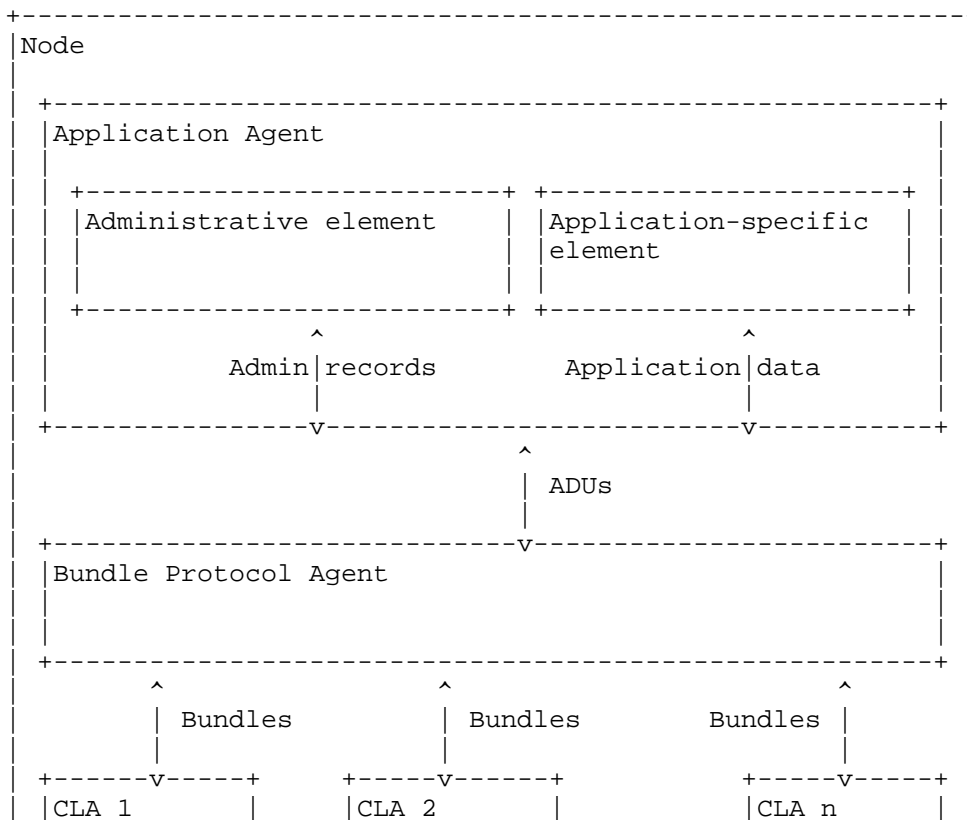
Block - A bundle protocol block is one of the protocol data structures that together constitute a well-formed bundle.

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; it is the content of the bundle's payload block. The terms "bundle content", "bundle payload", and "payload" are used interchangeably in this document.

Partial payload - A partial payload is a payload that comprises either the first N bytes or the last N bytes of some other payload of length M, such that $0 < N < M$.

Fragment - A fragment is a bundle whose payload block contains a partial payload.

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. Each bundle node has three conceptual components, defined below, as shown in Figure 2: a "bundle protocol agent", a set of zero or more "convergence layer adapters", and an "application agent".



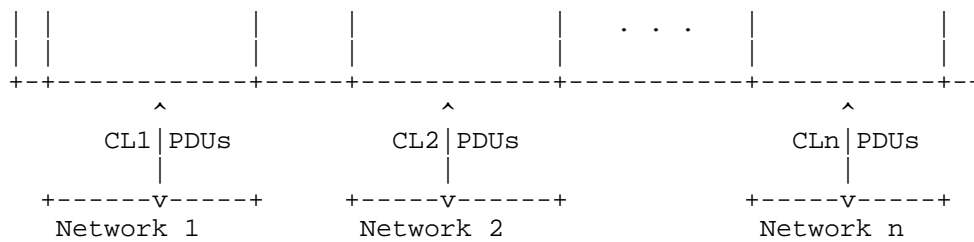


Figure 2: Components of a BP Node

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.

Convergence layer adapter - A convergence layer adapter (CLA) is a node component that sends and receives bundles on behalf of the BPA, utilizing the services of some 'native' protocol stack that is supported in one of the networks within which the node is functionally located.

Application agent - The application agent (AA) of a node is the node component that utilizes the BP services to effect communication for some user purpose. The application agent in turn has two elements, an administrative element and an application-specific element.

Application-specific element - The application-specific element of an AA is the node component that constructs, requests transmission of, accepts delivery of, and processes units of user application data.

Administrative element - The administrative element of an AA is the node component that constructs and requests transmission of administrative records (defined below), including status reports and custody signals, and accepts delivery of and processes any custody signals that the node receives.

Administrative record - A BP administrative record is an application data unit that is exchanged between the administrative elements of nodes' application agents for some BP administrative purpose. The formats of some fundamental administrative records (and of no other application data units) are defined in this specification.

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 common identifier, called a "bundle endpoint ID"

(or, in this document, simply "endpoint ID"; endpoint IDs are described in detail in Section 4.4.1 below).

Singleton endpoint - A singleton endpoint is an endpoint that always contains exactly one member.

Registration - A registration is the state machine characterizing a given node's membership in a given endpoint. Any single registration has an associated delivery failure action as defined below and must at any time be in one of two states: Active or Passive.

Delivery - 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 any relevant metadata (an implementation matter), has been presented to the node's application agent in a manner consistent with the state of that registration.

Deliverability - 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) the bundle has not yet been "abandoned" (as defined below) subject to this registration.

Abandonment - To abandon a bundle subject to some registration is to assert that the bundle is not deliverable subject to that registration.

Delivery failure action - The delivery failure action of a registration is the action that is to be taken when a bundle that is "deliverable" subject to that registration is received at a time when the registration is in the Passive state.

Destination - The destination of a bundle is the endpoint comprising the node(s) at which the bundle is to be delivered (as defined below).

Minimum reception group - The minimum reception group of an endpoint is the minimum number of members of the endpoint (nodes) at which the bundle must have been delivered in order for the bundle to be considered delivered to the endpoint.

Transmission - A transmission is an attempt by a node's BPA to cause copies of a bundle to be delivered at all nodes in the minimum reception group of some endpoint (the bundle's destination) in

response to a transmission request issued by the node's application agent.

Forwarding - To forward a bundle to a node is to invoke the services of a CLA in a sustained effort to cause a copy of the bundle to be received by that node.

Discarding - To discard a bundle is to cease all operations on the bundle and functionally erase all references to it. The specific procedures by which this is accomplished are an implementation matter.

Retention constraint - A retention constraint is an element of the state of a bundle that prevents the bundle from being discarded. That is, a bundle cannot be discarded while it has any retention constraints.

Deletion - To delete a bundle is to remove unconditionally all of the bundle's retention constraints, enabling the bundle to be discarded.

Custodian - A custodian of a bundle is a node that has determined that it will retain a copy of that bundle for an indefinite period of time, forwarding and possibly re-forwarding the bundle as appropriate, until it detects one of the conditions under which it may cease being a custodian of that bundle (discussed later).

Taking custody - To take custody of a bundle is to become a custodian of that bundle.

Accepting custody - To accept custody of a bundle is to take custody of the bundle, mark the bundle in such a way as to indicate this custodianship to nodes that subsequently receive copies of the bundle, and announce this custodianship to all current custodians of the bundle.

Refusing custody - To "refuse custody" of a bundle is to notify all current custodians of that bundle that an opportunity to take custody of the bundle has been declined.

Releasing custody - To release custody of a bundle is to cease to be a custodian of the bundle.

3.2. Discussion of BP concepts

Multiple instances of the same bundle (the same unit of DTN protocol data) might exist concurrently in different parts of a network --

possibly differing in some blocks -- 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 (copy), in that node's local memory, of some bundle that is in the network.

The 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 payload for a bundle forwarded in response to reception of a bundle is the payload of the received bundle.

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.

The manner in which the functions of the BPA are performed 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.

Every CLA implements its own thin layer of protocol, interposed between BP and the (usually "top") protocol(s) of the underlying native protocol stack; this "CL protocol" may only serve to multiplex and de-multiplex bundles to and from the underlying native protocol, or it may offer additional CL-specific functionality. The manner in which a CLA sends and receives bundles is, again, wholly an implementation matter. The definitions of CLAs and CL protocols are beyond the scope of this specification.

Note that the administrative element of a node's application agent may itself, in some cases, function as a convergence-layer adapter. That is, outgoing bundles may be "tunneled" through encapsulating bundles:

- . An outgoing bundle constitutes a byte array. This byte array may, like any other, be presented to the bundle protocol agent as an application data unit that is to be transmitted to some endpoint.
- . The original bundle thus forms the payload of an encapsulating bundle that is forwarded using some other convergence-layer protocol(s).

- . When the encapsulating bundle is received, its payload is delivered to the peer application agent administrative element, which then instructs the bundle protocol agent to dispatch that original bundle in the usual way.

The purposes for which this technique may be useful (such as cross-domain security) are beyond the scope of this specification.

The only interface between the BPA and the application-specific element of the AA is the BP service interface. But between the BPA and the administrative element of the AA there is a (conceptual) private control interface in addition to the BP service interface. This private control interface enables the BPA and the administrative element of the AA to direct each other to take action under specific circumstances

In the case of a node that serves simply as a BP "router", 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.

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.

The destination of every bundle is an endpoint, which may or may not be singleton. The source of every bundle is a node, identified by the endpoint ID for some singleton endpoint that contains that node.

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; or (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 typically be determined from the endpoint's ID. 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.

Any number of transmissions may be concurrently undertaken by the bundle protocol agent of a given node.

When the bundle protocol agent of a node determines that a bundle must be forwarded to a node (either to a node that is a member of the bundle's destination endpoint or to some intermediate forwarding node) in the course of completing the successful transmission of that bundle, it invokes the services of a CLA in a sustained effort to cause a copy of the bundle to be received by that node.

Upon reception, the processing of a bundle that has been received by 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 newly 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.

Whenever, for some implementation-specific reason, a node's BPA finds it impossible to immediately deliver a bundle that is deliverable, delivery of the bundle has failed. In this event, the delivery failure action associated with the applicable registration must be taken. Delivery failure action MUST be one of the following:

- . defer delivery 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 the Active state; or
- . abandon 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 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.

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 next bundle that is due for delivery according to the BPA's bundle delivery scheduling policy, an implementation matter.

Normally only registrations' registered delivery failure actions cause deliveries to be abandoned.

Custody of a bundle MAY be taken only if the destination of the bundle is a singleton endpoint. Custody MAY be released only 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; (c) the current custodian chooses to fragment the bundle, releasing custody of the original bundle and taking custody of the fragments instead, or (d) the bundle is explicitly deleted for some reason, such as lifetime expiration.

The custody transfer mechanism in BP provides a means of recovering from data loss along the path to the destination node. When the custodian of a bundle forwards that bundle it SHOULD set a retransmission timer; reception of a responding custody signal of any kind prior to timer expiration MUST disable that timer. Upon expiration of that timer, the custodian MUST re-forward the bundle. When a bundle for which custody has been taken arrives at a node from which it must be forwarded, and that node determines that it will forward the bundle but will not take custody, the receiving node SHOULD send a "custody delegation" signal back to the custodian indicating the next node to which the bundle will be forwarded together with an estimate of the interval that will elapse between the time the bundle was received and the time at which it will be forwarded. This mechanism is intended to facilitate accurate timeout interval calculation for this bundle.

Computation of the timeout interval for a bundle's custody transfer timer (i.e., determination of the moment at which a responding custody signal is expected) is an implementation matter and may be dynamically responsive to changes in connectivity. In some environments it may be impossible to compute this interval with

operationally satisfactory accuracy; in such environments the use of custody transfer services is contraindicated.

Alternatively, when custody transfer for a given bundle is not requested, data loss along the path to the destination node can be minimized by utilizing reliable convergence-layer protocols between neighbors on all segments of the end-to-end path. This approach may make more efficient use of links than custody transfer because a convergence-layer protocol may perform finer-grained retransmission than custody transfer does, retransmitting only the specific portions of a transmitted bundle that were not received, rather than the entire bundle. However, in some environments there may be segments of the end-to-end path for which no reliable convergence-layer protocol is available; in such environments the use of reliable convergence-layer protocols wherever possible can reduce the incidence of data loss.

3.3. Services Offered by Bundle Protocol Agents

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

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

4. Bundle Format

The format of bundles SHALL conform to the Concise Binary Object Representation (CBOR [RFC7049]).

Each bundle SHALL be a concatenated sequence of at least two blocks, represented as a CBOR indefinite-length array. The first block in the sequence (the first item of the array) MUST be a primary bundle block in CBOR representation as described below; the bundle MUST have exactly one primary bundle block. The primary block MUST be followed by one or more canonical bundle blocks (additional array items) in CBOR representation as described below. The last such block MUST be a payload block; the bundle MUST have exactly one payload block. The last item of the array, immediately following the payload block, SHALL be a CBOR "break" stop code.

4.1. BP Fundamental Data Structures

4.1.1. CRC Type

CRC type is an unsigned integer type code for which the following values (and no others) are valid:

- . 0 indicates "no CRC is present."
- . 1 indicates "a CRC-16 (a.k.a., CRC-16-ANSI) is present."
- . 2 indicates "a standard IEEE 802.3 CRC-32 is present."

CRC type SHALL be represented as a CBOR unsigned integer.

4.1.2. CRC

CRC SHALL be omitted from a block if and only if the block's CRC type code is zero.

When not omitted, the CRC SHALL be represented as a CBOR unsigned integer.

4.1.3. Bundle Processing Control Flags

Bundle processing control flags assert properties of the bundle as a whole rather than of any particular block of the bundle. They are conveyed in the primary block of the bundle.

The following properties are asserted by the bundle processing control flags:

- . The bundle is a fragment. (Boolean)
- . The bundle's payload is an administrative record. (Boolean)
- . The bundle must not be fragmented. (Boolean)
- . Custody transfer requested for this bundle. (Boolean)
- . The bundle's destination endpoint is a singleton. (Boolean)
- . Acknowledgment by the user application is requested. (Boolean)
- . Status time is requested in all status reports. (Boolean)
- . The bundle contains a "manifest" extension block. (Boolean)
- . Flags requesting types of status reports (all Boolean):

- o Request reporting of bundle reception.
- o Request reporting of custody transfer request processing.
- o Request reporting of bundle forwarding.
- o Request reporting of bundle delivery.
- o Request reporting of bundle deletion.

If the bundle processing control flags indicate that the bundle's application data unit is an administrative record, then the custody transfer requested flag value must be zero and all status report request flag values must be zero.

If the custody transfer requested flag value is 1, then the source node is requesting that every receiving node accept custody of the bundle.

If the bundle's source node is omitted (i.e., the source node ID is the ID of the null endpoint, which has no members as discussed below; this option enables anonymous bundle transmission), 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 value must be zero, the "Bundle must not be fragmented" flag value must be 1, and all status report request flag values must be zero.

The bundle processing control flags SHALL be represented as a CBOR unsigned integer item containing a bit field of 16 bits indicating the control flag values as follows:

- . Bit 0 (the high-order bit, 0x8000): reserved.
- . Bit 1 (0x4000): reserved.
- . Bit 2 (0x2000): reserved.
- . Bit 3(0x1000): bundle deletion status reports are requested.
- . Bit 4(0x0800): bundle delivery status reports are requested.
- . Bit 5(0x0400): bundle forwarding status reports are requested.
- . Bit 6(0x0200): custody request processing status reports are requested.
- . Bit 7(0x0100): bundle reception status reports are requested.
- . Bit 8(0x0080): bundle contains a Manifest block.
- . Bit 9(0x0040): status time is requested in all status reports.
- . Bit 10(0x0020): user application acknowledgement is requested.
- . Bit 11(0x0010): destination is a singleton endpoint.
- . Bit 12(0x0008): custody transfer is requested.
- . Bit 13(0x0004): bundle must not be fragmented.

- . Bit 14(0x0002): payload is an administrative record.
- . Bit 15 (the low-order bit, 0x0001: bundle is a fragment.

4.1.4. Block Processing Control Flags

The block processing control flags assert properties of canonical bundle blocks. They are conveyed in the header of the block to which they pertain.

The following properties are asserted by the block processing control flags:

- . This block must be replicated in every fragment. (Boolean)
- . Status report must be transmitted if this block can't be processed. (Boolean)
- . Block must be removed from the bundle if it can't be processed. (Boolean)
- . Bundle must be deleted if this block can't be processed. (Boolean)

For each bundle whose bundle processing control flags indicate that the bundle's application data unit is an administrative record, or whose source node ID is the null endpoint ID as defined below, the value of the "Transmit status report if block can't be processed" flag in every canonical block of the bundle must be zero.

The block processing control flags SHALL be represented as a CBOR unsigned integer item containing a bit field of 8 bits indicating the control flag values as follows:

- . Bit 0 (the high-order bit, 0x80): reserved.
- . Bit 1 (0x40): reserved.
- . Bit 2(0x20): reserved.
- . Bit 3(0x10): reserved.
- . Bit 4(0x08): bundle must be deleted if block can't be processed.
- . Bit 5(0x04): status report must be transmitted if block can't be processed.
- . Bit 6(0x02): block must be removed from bundle if it can't be processed.
- . Bit 7(the low-order bit, 0x01): block must be replicated in every fragment.

4.1.5. Identifiers

4.1.5.1. Endpoint ID

The destinations of bundles are bundle endpoints, identified by text strings termed "endpoint IDs" (see Section 3.1). Each endpoint ID (EID) is 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" >

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.

Note that, although endpoint IDs are URIs, implementations of the BP service interface may support expression of endpoint IDs in some internationalized manner (e.g., Internationalized Resource Identifiers (IRIs); see [RFC3987]).

The endpoint ID "dtn:none" identifies the "null endpoint", the endpoint that by definition never has any members.

Each BP endpoint ID (EID) SHALL be represented as a CBOR array comprising a 2-tuple.

The first item of the array SHALL be the code number identifying the endpoint's URI scheme [URI], as defined in the registry of URI scheme code numbers for Bundle Protocol maintained by IANA as described in Section 9. [URIREG]. Each URI scheme code number SHALL be represented as a CBOR unsigned integer.

The second item of the array SHALL be the applicable CBOR representation of the scheme-specific part (SSP) of the EID, defined as follows:

- . If the EID's URI scheme is "dtn" then the SSP SHALL be represented as a CBOR text string unless the EID's SSP is "none", in which case the SSP SHALL be represented as a CBOR unsigned integer with the value zero.
- . If the EID's URI scheme is "ipn" then the SSP SHALL be represented as a CBOR array comprising a 2-tuple. The first item of this array SHALL be the EID's node number represented as a CBOR unsigned integer. The second item of this array

SHALL be the EID's service number represented as a CBOR unsigned integer.

- . Definitions of the CBOR representations of the SSPs of EIDs encoded in other URI schemes are included in the specifications defining those schemes.

4.1.5.2. Node ID

For many purposes of the Bundle Protocol it is important to identify the node that is operative in some context.

As discussed in 3.1 above, nodes are distinct from endpoints; specifically, an endpoint is a set of zero or more nodes. But rather than define a separate namespace for node identifiers, we instead use endpoint identifiers to identify nodes, subject to the following restrictions:

- . Every node MUST be a member of at least one singleton endpoint.
- . The EID of any singleton endpoint of which a node is a member MAY be used to identify that node. A "node ID" is an EID that is used in this way.
- . A node's membership in a given singleton endpoint MUST be sustained at least until the nominal operation of the Bundle Protocol no longer depends on the identification of that node using that endpoint's ID.

4.1.6. DTN Time

A DTN time is an unsigned integer indicating a count of seconds since the start of the year 2000 on the Coordinated Universal Time (UTC) scale [UTC]. Each DTN time SHALL be represented as a CBOR unsigned integer item.

4.1.7. Creation Timestamp

Each creation timestamp SHALL be represented as a CBOR array item comprising a 2-tuple.

The first item of the array SHALL be a DTN time.

The second item of the array SHALL be the creation timestamp's sequence number, represented as a CBOR unsigned integer.

4.1.8. Block-type-specific Data

Block-type-specific data in each block (other than the primary block) SHALL be the applicable CBOR representation of the content of

the block. Details of this representation are included in the specification defining the block type.

4.2. Bundle Representation

This section describes the primary block in detail and non-primary blocks in general. 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 [BPSEC]) may require that BP implementations conforming to those protocols construct and process additional blocks.

4.2.1. Bundle

Each bundle SHALL be represented as a CBOR indefinite-length array. The first item of this array SHALL be the CBOR representation of a Primary Block. Every other item of the array except the last SHALL be the CBOR representation of a Canonical Block. The last item of the array SHALL be a CBOR "break" stop code.

4.2.2. Primary Bundle Block

The primary bundle block contains the basic information needed to forward bundles to their destinations.

Each primary block SHALL be represented as a CBOR array; the number of elements in the array SHALL be 8 (if the bundle is not a fragment and CRC type is zero) or 9 (if the bundle is not a fragment and CRC type is non-zero) or 10 (if the bundle is a fragment and CRC type is zero) or 11 (if the bundle is a fragment and CRC-type is non-zero).

The fields of the primary bundle block SHALL be as follows, listed in the order in which they MUST appear:

Version: An unsigned integer value indicating the version of the bundle protocol that constructed this block. The present document describes version 7 of the bundle protocol. Version number SHALL be represented as a CBOR unsigned integer item.

Bundle Processing Control Flags: The Bundle Processing Control Flags are discussed in Section 4.1.3. above.

CRC Type: CRC Type codes are discussed in Section 4.1.1. above.

Destination EID: The Destination EID field identifies the bundle endpoint that is the bundle's destination, i.e., the endpoint that contains the node(s) at which the bundle is to be delivered.

Source node ID: The Source node ID field identifies the bundle node at which the bundle was initially transmitted, except that Source node ID may be the null endpoint ID in the event that the bundle's source chooses to remain anonymous.

Report-to EID: The Report-to EID field identifies the bundle endpoint to which status reports pertaining to the forwarding and delivery of this bundle are to be transmitted.

Creation Timestamp: The creation timestamp is a pair of unsigned integers that, together with the source node ID and (if the bundle is a fragment) the fragment offset, serve to identify the bundle. The first of these integers is the bundle's creation time, while the second is the bundle's creation timestamp sequence number. Bundle creation time shall be 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 shall be 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. For nodes that lack accurate clocks (that is, nodes that are not at all moments able to determine the current UTC time to within 30 seconds), bundle creation time MUST be set to zero and the counter used as the source of the bundle sequence count MUST NEVER be reset to zero. In any case, a source Bundle Protocol Agent MUST NEVER create two distinct bundles with the same source node ID and bundle creation timestamp. The combination of source node ID and bundle creation timestamp serves to identify a single transmission request, enabling it to be acknowledged by the receiving application (provided the source node ID is not the null endpoint ID).

Lifetime: The lifetime field is an unsigned integer 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 a bundle's age exceeds its lifetime, bundle nodes need no longer retain or forward the bundle; the bundle SHOULD be deleted from the network. Bundle lifetime SHALL be represented as a CBOR unsigned integer item.

Fragment offset: If and only if the Bundle Processing Control Flags of this Primary block indicate that the bundle is a fragment,

fragment offset SHALL be present in the primary block. Fragment offset SHALL be represented as a CBOR unsigned integer indicating the offset from the start of the original application data unit at which the bytes comprising the payload of this bundle were located.

Total Application Data Unit Length: If and only if the Bundle Processing Control Flags of this Primary block indicate that the bundle is a fragment, total application data unit length SHALL be present in the primary block. Total application data unit length SHALL be represented as a CBOR unsigned integer indicating the total length of the original application data unit of which this bundle's payload is a part.

CRC: If and only if the value of the CRC type field of this Primary block is non-zero, a CRC SHALL be present in the primary block. The length and nature of the CRC SHALL be as indicated by the CRC type. The CRC SHALL be computed over the concatenation of all bytes of the primary block including the CRC field itself, which for this purpose SHALL be temporarily populated with the value zero.

4.2.3. Canonical Bundle Block Format

Every block other than the primary block (which blocks are termed "canonical" blocks) SHALL be represented as a CBOR array; the number of elements in the array SHALL be 6 (if CRC type is zero) or 7 (otherwise).

The fields of every canonical block SHALL be as follows, listed in the order in which they MUST appear:

- . Block type code, an unsigned integer. Bundle block type code 1 indicates that the block is a bundle payload block. Block type codes 2 through 9 are explicitly reserved as noted later in this specification. Block type codes 192 through 255 are not reserved and are available for private and/or experimental use. All other block type code values are reserved for future use.
- . Block number, an unsigned integer. The block number uniquely identifies the block within the bundle, enabling blocks (notably bundle security protocol blocks) to explicitly reference other blocks in the same bundle. Block numbers need not be in continuous sequence, and blocks need not appear in block number sequence in the bundle. The block number of the payload block is always zero.
- . Block processing control flags as discussed in Section 4.1.4 above.
- . CRC type as discussed in Section 4.1.1 above.

- . If and only if the value of the CRC type field of this block is non-zero, a CRC. If present, the length and nature of the CRC SHALL be as indicated by the CRC type and the CRC SHALL be computed over the concatenation of all bytes of the block including the CRC field itself, which for this purpose SHALL be temporarily populated with the value zero.
- . Block data length, an unsigned integer. The block data length field SHALL contain the aggregate length of all remaining fields of the block, i.e., the block-type-specific data fields. Block data length SHALL be represented as a CBOR unsigned integer item.
- . Block-type-specific data fields, whose nature and order are type-specific and whose aggregate length in octets is the value of the block data length field. For the Payload Block in particular (block type 1), there SHALL be exactly one block-type-specific data field, termed the "payload", which SHALL be an application data unit, or some contiguous extent thereof, represented as a CBOR byte string.

4.3. Extension Blocks

"Extension blocks" are all blocks other than the primary and payload blocks. Because not all extension blocks are 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 that the node cannot process. The values of the block processing control flags indicate the action to be taken by the bundle protocol agent when this is the case.

(Note that, while CBOR permits considerable flexibility in the encoding of bundles, this flexibility must not be interpreted as inviting increased complexity in protocol data unit structure.)

The following extension blocks are defined in other DTN protocol specification documents as noted:

- . Block Integrity Block (block type 2) and Block Confidentiality Block (block type 3) are defined in the Bundle Security Protocol specification (work in progress).
- . Manifest Block (block type 4) is defined in the Manifest Extension Block specification (TBD). The manifest block identifies the blocks that were present in the bundle at the time it was created. The bundle MUST contain one (1) occurrence of this type of block if the value of the "manifest"

- flag in the bundle processing control flags is 1; otherwise the bundle MUST NOT contain any Manifest block.
- . The Flow Label Block (block type 6) is defined in the Flow Label Extension Block specification (TBD). The flow label block is intended to govern transmission of the bundle by convergence-layer adapters.

The following extension blocks are defined in the current document.

4.3.1. Current Custodian

The Current Custodian block, block type 5, identifies a node that is known to have accepted custody of the bundle. The block-type-specific data of this block is the node ID of a custodian, which SHALL take the form of an endpoint ID represented as described in Section 4.1.5.2. above. The bundle MAY contain one or more occurrences of this type of block.

4.3.2. Previous Node

The Previous Node block, block type 7, identifies the node that forwarded this bundle to the local node (i.e., to the node at which the bundle currently resides); its block-type-specific data is the node ID of that forwarder node which SHALL take the form of a node ID represented as described in Section 4.1.5.2. above. If the local node is the source of the bundle, then the bundle MUST NOT contain any previous node block. Otherwise the bundle MUST contain one (1) occurrence of this type of block. If present, the previous node block MUST be the FIRST block following the primary block, as the processing of other extension blocks may depend on its value.

4.3.3. Bundle Age

The Bundle Age block, block type 8, contains the number of seconds that have elapsed between the time the bundle was created and time at which it was most recently forwarded. It is intended for use by nodes lacking access to an accurate clock, to aid in determining the time at which a bundle's lifetime expires. The block-type-specific data of this block is an unsigned integer containing the age of the bundle in seconds, which SHALL be represented as a CBOR unsigned integer item. (The age of the bundle is the sum of all known intervals of the bundle's residence at forwarding nodes, up to the time at which the bundle was most recently forwarded, plus the summation of signal propagation time over all episodes of transmission between forwarding nodes. Determination of these values is an implementation matter.) If the bundle's creation time is zero, then the bundle MUST contain exactly one (1) occurrence of

this type of block; otherwise, the bundle MAY contain at most one (1) occurrence of this type of block.

4.3.4. Hop Count

The Hop Count block, block type 9, contains two unsigned integers, hop limit and hop count. A "hop" is here defined as an occasion on which a bundle was forwarded from one node to another node. The hop count block is mainly intended as a safety mechanism, a means of identifying bundles for removal from the network that can never be delivered due to a persistent forwarding error: a bundle SHOULD be deleted when its hop count exceeds its hop limit. Procedures for determining the appropriate hop limit for a block are beyond the scope of this specification. The block-type-specific data in a hop count block SHALL be represented as a CBOR array comprising a 2-tuple. The first item of this array SHALL be the bundle's hop limit, represented as a CBOR unsigned integer. The second item of this array SHALL be the bundle's hop count, represented as a CBOR unsigned integer. A bundle MAY contain at most one (1) occurrence of this type of block.

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. Supplementary DTN protocol specifications (including, but not restricted to, the Bundle Security Protocol [BPSEC]) may augment, override, or supersede the mandates of this document.

5.1. Generation of Administrative Records

All transmission of bundles is in response to bundle transmission requests presented by nodes' application agents. When required to "generate" an administrative record (such as 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 two cases:

- . the reception of a bundle containing at least one block that cannot be processed, for which the value of the "transmit status report if block could not be processed" block processing flag is 1, and
- . 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 transfer status report" is a bundle status report with the "reporting node attempted custody transfer" 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 "current custodian" of a bundle is a node identified in a Current Custodian extension block of that bundle.

5.2. Bundle Transmission

The steps in processing a bundle transmission request are:

Step 1: If custody transfer is requested for this bundle transmission then the destination MUST be a singleton endpoint. If, moreover, custody acceptance by the source node is required when custody is requested (an implementation matter) but the conditions under which custody of the bundle may be accepted are not satisfied, then the request cannot be honored and all remaining steps of this procedure MUST be skipped.

Step 2: Transmission of the bundle is initiated. An outbound bundle MUST be created per the parameters of the bundle transmission request, with the retention constraint "Dispatch pending". The source node ID of the bundle MUST be either the null endpoint ID, indicating that the source of the bundle is anonymous, or else the EID of a singleton endpoint whose only member is the node of which the BPA is a component.

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 4. In particular:

- . The bundle protocol agent MUST determine which node(s) to forward the bundle to. The bundle protocol agent MAY choose either to forward the bundle directly to its destination node(s) (if possible) or to forward the bundle to some other node(s) for further forwarding. The manner in which this decision is made may depend on the scheme name in the destination endpoint ID and/or on other state but in any case is beyond the scope of this document. If the BPA elects to forward the bundle to some other node(s) for further forwarding but finds it impossible to select any node(s) to forward the bundle to, then forwarding is contraindicated.
- . Provided the bundle protocol agent succeeded in selecting the node(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 those nodes. The

manner in which specific appropriate convergence layer adapters are selected is beyond the scope of this document. If the agent finds it impossible to select any appropriate convergence layer adapter(s) 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 4, 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 MUST be followed.

Step 5: For each node 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 that node. Determining the time at which the bundle protocol agent invokes convergence layer adapter services is a BPA implementation matter. Determining the time at which each convergence layer adapter subsequently responds to this service invocation by sending the bundle is a convergence-layer adapter implementation matter. Note that:

- . If the bundle contains a flow label extension block then that flow label value MAY identify procedures for determining the order in which convergence layer adapters must send bundles, e.g., considering bundle source when determining the order in which bundles are sent. The definition of such procedures is beyond the scope of this specification.
- . If the bundle has a bundle age block, then at the last possible moment before the CLA initiates conveyance of the bundle node via the CL protocol the bundle age value MUST be increased by the difference between the current time and the time at which the bundle was received (or, if the local node is the source of the bundle, created).

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 to which the bundle was forwarded 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 are:

Step 1: The bundle protocol agent MUST determine whether or not to declare failure in forwarding the bundle. 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 Section 5.4.2 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 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. The bundle protocol agent MUST handle the custody transfer failure by generating a custody signal of type 1 (custody refusal) for the bundle, destined for the bundle's current custodian(s); 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 each current custodian still has a copy.) In addition, if the "request reporting of custody transfer attempted" flag in the bundle's status report request field is set to 1, a custody transfer status report with the same reason code SHOULD be generated, destined for the report-to endpoint ID of the bundle.

If the bundle's custody transfer requested flag (in the bundle processing flags field) is set to 0, then the bundle protocol agent MAY forward the bundle back to the node that sent it, as identified by the Previous Node block.

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 Section 5.14 MUST be followed, citing the reason for which forwarding was determined to be contraindicated.

5.5. Bundle Expiration

A bundle expires when the bundle's age exceeds its lifetime as specified in the primary bundle block. Bundle age MAY be determined by subtracting the bundle's creation timestamp time from the current time if (a) that timestamp time is not zero and (b) the local node's clock is known to be accurate (as discussed in section 4.5.1 above); otherwise bundle age MUST be obtained from the Bundle Age extension 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 5.14 MUST be followed.

5.6. Bundle Reception

The steps in processing a bundle that has been 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 Section 5.14 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 processing continues with the next extension block that the bundle protocol agent cannot process, if any; otherwise, processing proceeds from step 4.

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 node ID, creation timestamp, and (if the bundle is a fragment) fragment offset as another bundle that (a) has not been discarded and (b) currently has the retention constraint "Custody accepted", then custody transfer redundancy MUST be handled; otherwise, processing proceeds from Step 5. The bundle protocol agent MUST handle custody transfer redundancy by generating a custody signal of type 1 (custody refusal) 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 MUST be 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. The bundle protocol agent MUST report custodial delivery by generating a custody signal of type 0 (custody acceptance) for the bundle, destined for the bundle's current custodian(s).

5.8. Bundle Fragmentation

It may at times be advantageous for bundle protocol agents to reduce the sizes of bundles in order to forward them. This might be the case, for example, if a node 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 node 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 that has any Current Custodian extension block citing any node other than the local node MUST NOT be fragmented. This restriction aside, 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 fragmented bundle (that is, the bundle that is being fragmented). Note that the payloads of fragments resulting from different fragmentation episodes, in different parts of the network, may be overlapping subsets of the fragmented bundle's payload.
- . The primary block of each fragment MUST differ from that of the fragmented bundle, in that the bundle processing flags of the fragment MUST indicate that the bundle is a fragment and both fragment offset and total application data unit length must be provided. Additionally, the CRC of the fragmented bundle, if any, MUST be replaced in each fragment by a new CRC computed for the primary block of that fragment.
- . The payload blocks of fragments will differ from that of the fragmented bundle as noted above.
- . If the fragmented bundle is not a fragment or is the fragment with offset zero, then all extension blocks of the fragmented bundle MUST be replicated in the fragment whose offset is zero.
- . Each of the fragmented bundle's extension blocks whose "Block must be replicated in every fragment" flag is set to 1 MUST be replicated in every fragment.
- . Beyond these rules, replication of extension blocks in the fragments is an implementation matter.
- . If the local node is a custodian of the fragmented bundle, then the BPA MUST release custody of the fragmented bundle before fragmentation occurs and MUST take custody of every fragment.

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 node 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 BPA MUST take custody of each fragmentary bundle whose payload is a subset of the reassembled application data unit,

- for which custody transfer is requested but the BPA has not yet taken custody.
- . The BPA MUST then release custody of every fragment whose payload is a subset of the reassembled application data unit, for which it has taken custody.
- . 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 the nodes that are members of destination endpoints as necessary; an application data unit MAY also be reassembled at some other node on the path to the destination.

5.10. Custody Transfer

The decision as to whether or not to accept custody of a bundle is an implementation matter that may involve both resource and policy considerations.

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 are defined as follows.

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

If the "request reporting of custody transfer attempted" flag in the bundle's status report request field is set to 1, a custody transfer status report with reason code 0 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 2 of Section 5.6) but has not yet been transmitted, then this report SHOULD be generated by simply turning on the "Reporting node attempted custody transfer" flag in that earlier report.

The bundle protocol agent MUST generate a custody signal of type 0 (custody acceptance) for the bundle, destined for the bundle's current custodian(s).

The bundle protocol agent MUST assert the new current custodian for the bundle. It does so by deleting all of the bundle's existing

Current Custodian extension blocks and inserting a new Current Custodian extension block whose value is the node ID of the local node.

If the value of a custody transfer timer interval for this bundle can be calculated with operationally satisfactory accuracy, then the bundle protocol agent SHOULD set a custody transfer countdown timer for the 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 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

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

5.11. Custody Transfer Success

Upon receipt of a custody signal of type 0 (custody acceptance) at a node that is a custodial node of the bundle identified in the custody signal, custody of the bundle MUST be released as described in Section 5.10.2.

5.12. Custody Transfer Failure

Custody transfer is determined to have failed at a custodial node for a given bundle when either (a) that node's custody transfer timer for that bundle (if any) expires or (b) a custody signal of type 1 (custody refusal) for that bundle is received at that node.

Upon determination of custody transfer failure due to expiration of a custody transfer countdown timer, the bundle protocol agent MUST re-forward the bundle, possibly on a different route (Section 5.4).

Upon determination of custody transfer failure due to reception of a custody signal of type 1 (custody refusal), the action taken by the bundle protocol agent is implementation-specific and may depend on the reason code cited for the refusal. For example, if the custody signal's reason code was "Depleted storage", the bundle protocol agent might choose to re-forward the bundle, possibly on a different route (Section 5.4). If the reason code was "Redundant reception",

on the other hand, this 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. Custody Transfer Deferral

Upon receipt of a bundle for which custody transfer retransmission service has been requested, which the bundle protocol agent plans to forward but for which it elects not to accept custody, the bundle protocol agent SHOULD generate a custody signal of type 2 (custody delegation) for the bundle, destined for the bundle's current custodian(s).

Custody transfer is determined to have been deferred at a custodial node for given bundle when a custody signal of type 2 (custody delegation) for that bundle is received at that node. The action taken by the bundle protocol agent in this event is implementation-specific. Notionally, this is an opportunity for the bundle protocol agent to revise its retransmission timeout interval for this bundle, based on the information provided in the custody signal: the next candidate custodian for the bundle is now known, and the minimum length of time before a custody acceptance signal will arrive can now be adjusted accordingly.

5.14. Bundle Deletion

The steps in deleting a bundle are:

Step 1: If the retention constraint "Custody accepted" currently prevents this bundle from being discarded, then:

- . Custody of the bundle 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.

5.15. Discarding a Bundle

As soon as a bundle has no remaining retention constraints it MAY be discarded, thereby releasing any persistent storage that may have been allocated to it.

5.16. 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 cancelled". For this purpose, the procedure defined in Section 5.14 MUST be followed.

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. Note that additional types of administrative records may be defined by supplementary DTN protocol specification documents.

Every administrative record consists of:

- . Record type code (an unsigned integer for which valid values are as defined below).
- . Record content in type-specific format.

Valid administrative record type codes are defined as follows:

+-----+-----+-----+-----+-----+-----+-----+															
	Value		Meaning												
+=====+=====+=====+=====+=====+=====+=====+															
	1		Bundle status report.												
+-----+-----+-----+-----+-----+-----+-----+															
	2		Custody signal.												
+-----+-----+-----+-----+-----+-----+-----+															

```
| (other) | Reserved for future use. |
+-----+-----+-----+-----+-----+
```

Figure 3: Administrative Record Type Codes

Each BP administrative record SHALL be represented as a CBOR array comprising a 2-tuple.

The first item of the array SHALL be a record type code, which SHALL be represented as a CBOR unsigned integer.

The second element of this array SHALL be the applicable CBOR representation of the content of the record. Details of the CBOR representation of administrative record types 1 and 2 are provided below. Details of the CBOR representation of other types of administrative record type are included in the specifications defining those records.

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.

Each bundle status report SHALL be represented as a CBOR array. The number of elements in the array SHALL be either 6 (if the subject bundle is a fragment) or 4 (otherwise).

The first item of the bundle status report array SHALL be bundle status information represented as a CBOR array of 5 elements. The five items of the bundle status information array shall provide information on the following five status assertions, in this order:

- . Reporting node received bundle.
- . Reporting node attempted custody transfer. When this status is asserted, a reason code value of 0 ("No additional information") SHALL indicate that custody was accepted.
- . Reporting node forwarded the bundle.
- . Reporting node delivered the bundle.
- . Reporting node deleted the bundle.

Each item of the bundle status information array SHALL be a bundle status item represented as a CBOR array; the number of elements in each such array SHALL be either 2 (if the value of the first item of this bundle status item is 1 AND the "Report status time" flag was set to 1 in the bundle processing flags of the bundle whose status is being reported) or 1 (otherwise). The first item of the bundle status item array SHALL be a status indicator, a Boolean value indicating whether or not the corresponding bundle status is asserted, represented as a CBOR Boolean value. The second item of the bundle status item array, if present, SHALL indicate the time (as reported by the local system clock, an implementation matter) at which the indicated status was asserted for this bundle, represented as a DTN time as described in Section 4.1.6. above.

The second item of the bundle status report array SHALL be the bundle status report reason code explaining the value of the status indicator, represented as a CBOR unsigned integer. Valid status report reason codes are defined in Figure 4 below but 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 [BPSEC]) may define additional reason codes.

+-----+-----+-----+-----+-----+-----+		
Value		Meaning
+=====+=====+=====+=====+=====+=====+		
0		No additional information.
+-----+-----+-----+-----+-----+-----+		
1		Lifetime expired.
+-----+-----+-----+-----+-----+-----+		
2		Forwarded over unidirectional link.
+-----+-----+-----+-----+-----+-----+		
3		Transmission canceled.
+-----+-----+-----+-----+-----+-----+		
4		Depleted storage.

Each custody signal SHALL be represented as a CBOR array. The number of elements in the array SHALL be 6 (if the subject bundle is a fragment) or 4 (otherwise).

The first item of the custody signal array SHALL be a signal type code represented as a CBOR unsigned integer. Valid custody signal types are defined as follows:

+-----+-----+-----+		
Value	Meaning	
+=====+=====+=====+		
0	Custody acceptance. The reporting node	
	accepted custody of the bundle.	
+-----+-----+-----+		
1	Custody refusal. The reporting node	
	refused custody of the bundle.	
+-----+-----+-----+		
2	Custody delegation: the bundle will be	
	forwarded but custody was not taken.	
+-----+-----+-----+		
(other)	Reserved for future use.	
+-----+-----+-----+		

Figure 5: Custody Signal Type Codes

The second item of the custody signal array SHALL be additional information amplifying the signal type code, represented as a CBOR array. The number of elements in the array SHALL be either 2 (if the signal type is 2) or 1 (otherwise).

When signal type is 0 or 1, the sole item in the additional information array SHALL be a custody signal reason code, represented as a CBOR unsigned integer. Valid custody signal reason codes are defined as follows:

+-----+-----+-----+		
Value	Meaning	
+=====+=====+=====+		
0	No additional information.	
+-----+-----+-----+		
1	Reserved for future use.	
+-----+-----+-----+		
2	Reserved for future use.	
+-----+-----+-----+		
3	Redundant (reception by a node that is a	
	custodial node for this bundle).	
+-----+-----+-----+		
4	Depleted storage.	
+-----+-----+-----+		
5	Destination endpoint ID unintelligible.	
+-----+-----+-----+		
6	No known route destination from here.	
+-----+-----+-----+		
7	No timely contact with next node on route.	
+-----+-----+-----+		
8	Block unintelligible.	
+-----+-----+-----+		
(other)	Reserved for future use.	
+-----+-----+-----+		

Figure 6: Custody Signal Reason Codes

When signal type is 2, the first item in the additional information array SHALL be the node ID of the node to which the reporting node anticipates forwarding the bundle, represented as described in Section 4.1.5.2. above, and the second item in this array SHALL be an estimate of the number of seconds that will have elapsed since reception of the bundle before the anticipated forwarding begins, represented as a CBOR unsigned integer.

The third item of the custody signal array SHALL be the source node ID identifying the source of the bundle for which custodial activity is being reported, represented as described in Section 4.1.5.2. above.

The fourth item of the custody signal array SHALL be the creation timestamp of the bundle for which custodial activity is being reported, represented as described in Section 4.1.7. above.

The fifth item of the custody signal array SHALL be present if and only if the bundle for which custodial activity is being reported contained a fragment offset. If present, it SHALL be the subject bundle's fragment offset represented as a CBOR unsigned integer item.

The sixth item of the custody signal array SHALL be present if and only if the bundle for which custodial activity is being reported contained a fragment offset. If present, it SHALL be the length of the subject bundle's payload represented as a CBOR unsigned integer item.

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 contain the fragment offset and fragment length.

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 signal type zero (custody acceptance), 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 signal type 1 (custody refusal), the administrative element of the application agent MUST direct the bundle protocol agent to follow the custody transfer failure procedure in Section 5.12.

For each received custody signal that has signal type 2 (custody delegation), the administrative element of the application agent MUST direct the bundle protocol agent to follow the custody delegation procedure in Section 5.13.

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 a bundle node that is reachable via the convergence layer protocol;
- . delivering to the bundle protocol agent a bundle that was sent by a 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 [BPSEC]) may expect convergence layer adapters that serve BP implementations conforming to those protocols to provide additional services such as reporting on the transmission and/or reception progress of individual bundles (at completion and/or incrementally), retransmitting data that were lost in

transit, discarding bundle-conveying data units that the convergence layer protocol determines are corrupt or inauthentic, or reporting on the integrity and/or authenticity of delivered bundles.

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 [BPSEC]; an informative overview of this architecture is provided in [SECO].

The bundle protocol has been designed with the notion that it may 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 as soon as practicable.

Likewise, because of the potentially high 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 two 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.

To provide end-to-end bundle authenticity and integrity, the Block Integrity Block (BIB) is used. The BIB allows any security-enabled entity along the delivery path to ensure the integrity of the bundle's payload or any other block other than a Block Confidentiality Block.

To provide payload confidentiality, the use of the Block Confidentiality Block (BCB) is available. The bundle payload, or any other block aside from the primary block and the Bundle Security Protocol blocks, may be encrypted to provide end-to-end payload confidentiality/privacy.

Additionally, convergence-layer protocols that ensure authenticity of communication between adjacent nodes in BP network topology SHOULD be used where available, to minimize the ability of unauthenticated nodes to introduce inauthentic traffic into the network.

Bundle security MUST NOT be invalidated by forwarding nodes even though they themselves might not use the Bundle Security Protocol.

In particular, while blocks MAY be added to bundles transiting intermediate nodes, removal of blocks with the 'Discard block if it can't be processed' flag set in the block processing control flags 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" and "ipn" URI schemes have been provisionally registered by IANA. See <http://www.iana.org/assignments/uri-schemes.html> for the latest details.

Registries of URI scheme type numbers, extension block type numbers, and administrative record type numbers will be required.

10. References

10.1. Normative References

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

[RFC7049] Borman, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, October 2013.

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

[URIREG] Thaler, D., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", RFC 7595, BCP 35, June 2015.

10.2. Informative References

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

[BPSEC] Birrane, E., "Bundle Security Protocol Specification", Work In Progress, October 2015.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

[RFC5050] Scott, M. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, November 2007.

[SECO] Farrell, S., Symington, S., Weiss, H., and P. Lovell, "Delay-Tolerant Networking Security Overview", Work Progress, 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 "Journées systemes de reference spatio-temporels", 2004.

11. Acknowledgments

This work is freely adapted from [RFC5050], which was an effort of the Delay Tolerant Networking Research Group. The following DTNRG participants contributed significant technical material and/or inputs to that document: 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 Carnegie Mellon University, Stephen Farrell of Trinity College Dublin, Peter Lovell of SPARTA, Inc., Manikantan Ramadas of Ohio University, and Howard Weiss of SPARTA, Inc.

This document was prepared using 2-Word-v2.0.template.dot.

12. Significant Changes from RFC 5050

Points on which this draft significantly differs from RFC 5050 include the following:

- . Clarify the difference between transmission and forwarding.
- . Amplify discussion of custody transfer. Move current custodian to an extension block, of which there can be multiple occurrences (possible support for the MITRE idea of multiple concurrent custodians, from several years ago); define that block in this specification.
- . Introduce the concept of "node ID" as functionally distinct from endpoint ID, while having the same syntax.
- . Restructure primary block, making it immutable. Add optional CRC.
- . Add optional CRCs to non-primary blocks.
- . Add block ID number to canonical block format (to support streamlined BSP).
- . Add bundle age extension block, defined in this specification.
- . Add previous node extension block, defined in this specification.
- . Add flow label extension block, *not* defined in this specification.
- . Add manifest extension block, *not* defined in this specification.
- . Add hop count extension block, defined in this specification.
- . Clean up a conflict between fragmentation and custody transfer that Ed Birrane pointed out.

Appendix A.

For More Information

Please refer comments to dtn@ietf.org. The Delay Tolerant Networking Research Group (DTNRG) Web site is located at <http://www.dtnrg.org>.

Copyright (c) 2016 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

Appendix B.

CDDL expression

For informational purposes, Carsten Bormann has kindly provided an expression of the Bundle Protocol specification in the CBOR Data Definition Language (CDDL). That CDDL expression is presented below, somewhat edited by the authors. Note that wherever the CDDL expression is in disagreement with the textual representation of the BP specification presented in the earlier sections of this document, the textual representation rules.

```
start = bundle

dtn-time = uint

creation-timestamp = [dtn-time, sequence: uint]

eid-generic = [uri-code, SSP: any]

uri-code = uint

eid = eid-choice .within eid-generic

eid-choice /= [dtn-code, SSP: bytes]

dtn-code = 1 ; TBD

eid-choice /= [ipn-code, SSP: [nodenum: uint, servicenum: uint]]

ipn-code = 2 ; TBD

bundle-control-flags = uint .bits bundleflagbits

bundleflagbits = &(
    reserved: 15
    reserved: 14
    reserved: 13
    bundle-deletion-status-reports-are-requested: 12
    bundle-delivery-status-reports-are-requested: 11
    bundle-forwarding-status-reports-are-requested: 10
    custody-transfer-status-reports-are-requested: 9
```



```
    bundle-reception-status-reports-are-requested: 8
    bundle-contains-a-Manifest-block: 7
    status-time-is-requested-in-all-status-reports: 6
    user-application-acknowledgement-is-requested: 5
    destination-is-a-singleton-endpoint: 4
    custody-transfer-is-requested: 3
    bundle-must-not-be-fragmented: 2
    payload-is-an-administrative-record: 1
    bundle-is-a-fragment: 0
)

crc = uint

block-control-flags = uint .bits blockflagbits

blockflagbits = &(amp;
    reserved: 7
    reserved: 6
    reserved: 5
    reserved: 4
    bundle-must-be-deleted-if-block-cannot-be-processed: 3
    status-report-must-be-transmitted-if-block-cannot-be-processed: 2
    block-must-be-removed-from-bundle-if-it-cannot-be-processed: 1
    block-must-be-replicated-in-every-fragment: 0
)

bundle = [primary-block, *extension-block, payload-block]

primary-block = [
```

```
    version: 7,  
    bundle-control-flags,  
    crc-type: uint,  
    destination: eid,  
    source-node: eid,  
    report-to: eid,  
    creation-timestamp,  
    lifetime: uint,  
    ? fragment-offset: uint,  
    ? total-application-data-length: uint,  
    ? crc,  
]  
  
canonical-block-generic = [  
    block-type-code: uint,  
    canonical-block-common,  
    content: any  
]  
  
canonical-block-common = (  
    block-number: uint,  
    block-control-flags,  
    crc-type: uint,  
    ? crc,  
)
```

```
canonical-block = canonical-block-choice .within canonical-block-  
generic  
  
canonical-block-choice /= payload-block  
  
payload-block = [1, canonical-block-common, adu-extent: payload]  
  
payload = bytes / bytes .cbor admin-record  
  
canonical-block-choice /= extension-block  
  
extension-block = extension-block-choice .within canonical-block  
  
extension-block-choice /= current-custodian-block  
  
current-custodian-block = [5, canonical-block-common, eid]  
  
extension-block-choice /= previous-node-block  
  
previous-node-block = [7, canonical-block-common, eid]  
  
extension-block-choice /= bundle-age-block  
  
bundle-age-block = [8, canonical-block-common, bundle-age: uint]  
  
extension-block-choice /= hop-count-block  
  
hop-count-block = [9, canonical-block-common,  
                    [hop-limit: uint,  
                     hop-count: uint]]  
  
admin-record-generic = [record-type: uint, any]  
  
admin-record = admin-record-choice .within admin-record-generic  
  
admin-record-choice /= bundle-status-report  
  
bundle-status-report = [1, [bundle-status-information,  
                             bundle-status-reason: uint,  
                             admin-common]]  
  
admin-common = (
```

```
        source-node: eid,
        creation-timestamp,
        ? fragment-offset: uint,
        ? payload-length: uint)

bundle-status-information = [
    reporting-node-received-bundle: bundle-status-item,
    reporting-node-attempted-custody-transfer: bundle-status-item,
    reporting-node-forwarded-the-bundle: bundle-status-item,
    reporting-node-delivered-the-bundle: bundle-status-item,
    reporting-node-deleted-the-bundle: bundle-status-item,
]

bundle-status-item = (
    asserted: Boolean,
    ? time-of-assertion: dtn-time)

admin-record-choice /= custody-signal

custody-signal = [2, [custody-signal-type-code: uint,
    custody-signal-information,
    admin-common]]

custody-signal-information = custody-reason-code: uint / delegation-
information

delegation-information = (
    next-hop-node: eid,
    seconds-until-forwarding: uint)
```

Authors' Addresses

Scott Burleigh
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109-8099
US
Phone: +1 818 393 3353
Email: Scott.Burleigh@jpl.nasa.gov

Kevin Fall
Carnegie Mellon University / Software Engineering Institute
4500 Fifth Avenue
Pittsburgh, PA 15213
US
Phone: +1 412 268 3304
Email: kfall@cmu.edu

Edward J. Birrane
Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd
Laurel, MD 20723
US
Phone: +1 443 778 7423
Email: Edward.Birrane@jhuapl.edu

Delay-Tolerant Networking Working Group
Internet Draft
Intended status: Standards Track
Expires: July 29, 2021

S. Burleigh
JPL, Calif. Inst. Of Technology
K. Fall
Roland Computing Services
E. Birrane
APL, Johns Hopkins University
January 25, 2021

Bundle Protocol Version 7
draft-ietf-dtn-bpbis-31.txt

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), 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/lid-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 July 29, 2021.

Copyright Notice

Copyright (c) 2021 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 Simplified BSD License.

Abstract

This Internet Draft presents a specification for the Bundle Protocol, adapted from the experimental Bundle Protocol specification developed by the Delay-Tolerant Networking Research group of the Internet Research Task Force and documented in RFC 5050.

Table of Contents

1. Introduction.....	3
2. Conventions used in this document.....	5
3. Service Description.....	5
3.1. Definitions.....	5
3.2. Discussion of BP concepts.....	9
3.3. Services Offered by Bundle Protocol Agents.....	12
4. Bundle Format.....	13
4.1. Bundle Structure.....	13
4.2. BP Fundamental Data Structures.....	14
4.2.1. CRC Type.....	14
4.2.2. CRC.....	14
4.2.3. Bundle Processing Control Flags.....	15
4.2.4. Block Processing Control Flags.....	16
4.2.5. Identifiers.....	17
4.2.5.1. Endpoint ID.....	17
4.2.5.1.1. The "dtn" URI scheme.....	18
4.2.5.1.2. The "ipn" URI scheme.....	20
4.2.5.2. Node ID.....	22
4.2.6. DTN Time.....	22
4.2.7. Creation Timestamp.....	22
4.2.8. Block-type-specific Data.....	23
4.3. Block Structures.....	23
4.3.1. Primary Bundle Block.....	23
4.3.2. Canonical Bundle Block Format.....	26
4.4. Extension Blocks.....	27
4.4.1. Previous Node.....	27
4.4.2. Bundle Age.....	28
4.4.3. Hop Count.....	28
5. Bundle Processing.....	29
5.1. Generation of Administrative Records.....	29
5.2. Bundle Transmission.....	30
5.3. Bundle Dispatching.....	30
5.4. Bundle Forwarding.....	30

5.4.1. Forwarding Contraindicated.....	33
5.4.2. Forwarding Failed.....	33
5.5. Bundle Expiration.....	33
5.6. Bundle Reception.....	34
5.7. Local Bundle Delivery.....	35
5.8. Bundle Fragmentation.....	36
5.9. Application Data Unit Reassembly.....	37
5.10. Bundle Deletion.....	38
5.11. Discarding a Bundle.....	38
5.12. Canceling a Transmission.....	38
6. Administrative Record Processing.....	38
6.1. Administrative Records.....	38
6.1.1. Bundle Status Reports.....	39
6.2. Generation of Administrative Records.....	42
7. Services Required of the Convergence Layer.....	43
7.1. The Convergence Layer.....	43
7.2. Summary of Convergence Layer Services.....	43
8. Implementation Status.....	44
9. Security Considerations.....	45
10. IANA Considerations.....	47
10.1. Bundle Block Types.....	47
10.2. Primary Bundle Protocol Version.....	48
10.3. Bundle Processing Control Flags.....	49
10.4. Block Processing Control Flags.....	51
10.5. Bundle Status Report Reason Codes.....	52
10.6. Bundle Protocol URI scheme types.....	53
10.7. URI scheme "dtn".....	54
10.8. URI scheme "ipn".....	55
11. References.....	56
11.1. Normative References.....	56
11.2. Informative References.....	56
12. Acknowledgments.....	57
13. Significant Changes from RFC 5050.....	58
Appendix A. For More Information.....	59
Appendix B. CDDL expression.....	60

1. Introduction

Since the publication of the Bundle Protocol Specification (Experimental RFC 5050 [RFC5050]) in 2007, the Delay-Tolerant Networking (DTN) Bundle Protocol has been implemented in multiple programming languages and deployed to a wide variety of computing platforms. This implementation and deployment experience has identified opportunities for making the protocol simpler, more capable, and easier to use. The present document, standardizing the Bundle Protocol (BP), is adapted from RFC 5050 in that context,

reflecting lessons learned. Significant changes from the Bundle Protocol specification defined in RFC 5050 are listed in section 13.

This document describes version 7 of BP.

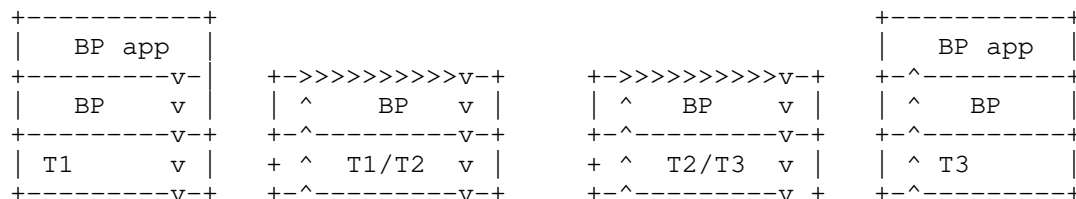
Delay Tolerant Networking is a network architecture providing communications in and/or through highly stressed environments. Stressed networking environments include those with intermittent connectivity, large and/or variable delays, and high bit error rates. To provide its services, BP may be viewed as sitting at the application layer of some number of constituent networks, forming a store-carry-forward overlay network. Key capabilities of BP include:

- . Ability to use physical motility for the movement of data
- . Ability to move the responsibility for error control from one node to another
- . Ability to cope with intermittent connectivity, including cases where the sender and receiver are not concurrently present in the network
- . Ability to take advantage of scheduled, predicted, and opportunistic connectivity, whether bidirectional or unidirectional, in addition to continuous connectivity
- . Late binding of overlay network endpoint identifiers to underlying constituent network addresses

For descriptions of these capabilities and the rationale for the DTN architecture, see [ARCH] and [SIGC].

BP's location within the standard protocol stack is as shown in Figure 1. BP uses underlying "native" transport and/or network protocols for communications within a given constituent network. The layer at which those underlying protocols are located is here termed the "convergence layer" and the interface between the bundle protocol and a specific underlying protocol is termed a "convergence layer adapter".

Figure 1 shows three distinct transport and network protocols (denoted T1/N1, T2/N2, and T3/N3).



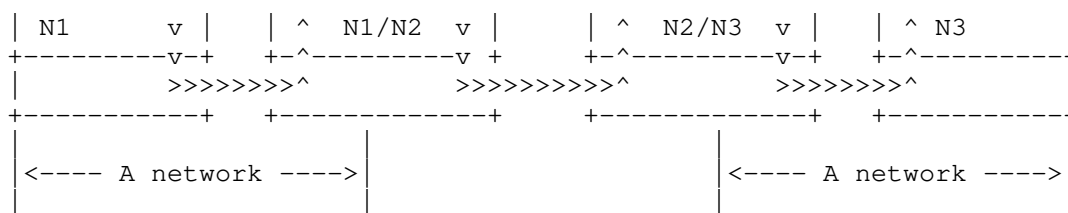


Figure 1: The Bundle Protocol in the Protocol Stack Model

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 internets. (However, the document does discuss the services that must be provided by each adapter at the convergence layer.)
- . The bundle route computation algorithm.
- . Mechanisms for populating the routing or forwarding information bases of bundle nodes.
- . The mechanisms for securing bundles en route.
- . The mechanisms for managing bundle nodes.

Note that implementations of the specification presented in this document will not be interoperable with implementations of RFC 5050.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Service Description

3.1. Definitions

Bundle - A bundle is a protocol data unit of BP, so named because negotiation of the parameters of a data exchange may be impractical in a delay-tolerant network: it is often better practice to "bundle" with a unit of application data all metadata that might be needed in order to make the data immediately usable when delivered to the

application. Each bundle comprises a sequence of two or more "blocks" of protocol data, which serve various purposes.

Block - A bundle protocol block is one of the protocol data structures that together constitute a well-formed bundle.

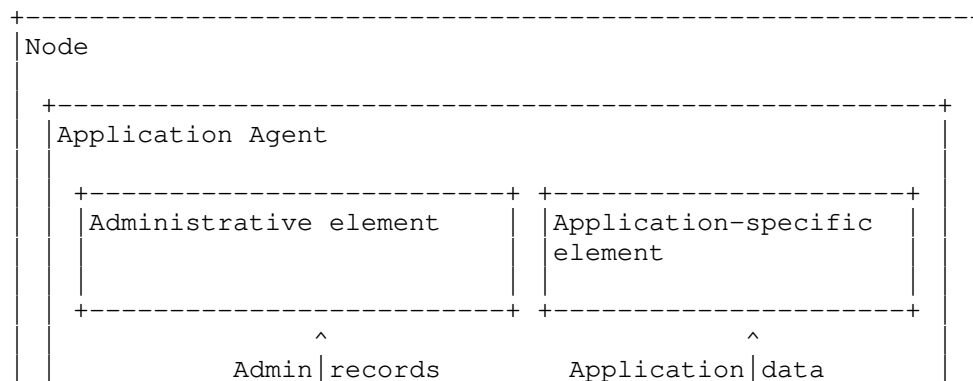
Application Data Unit (ADU) - An application data unit is the unit of data whose conveyance to the bundle's destination is the purpose for the transmission of some bundle that is not a fragment (as defined below).

Bundle payload - A bundle payload (or simply "payload") is the content of the bundle's payload block. The terms "bundle content", "bundle payload", and "payload" are used interchangeably in this document. For a bundle that is not a fragment (as defined below), the payload is an application data unit.

Partial payload - A partial payload is a payload that comprises either the first N bytes or the last N bytes of some other payload of length M, such that $0 < N < M$. Note that every partial payload is a payload and therefore can be further subdivided into partial payloads.

Fragment - A fragment, a.k.a. "fragmentary bundle", is a bundle whose payload block contains a partial payload.

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. Each bundle node has three conceptual components, defined below, as shown in Figure 2: a "bundle protocol agent", a set of zero or more "convergence layer adapters", and an "application agent". ("CL1 PDUs" are the PDUs of the convergence-layer protocol used in network 1.)



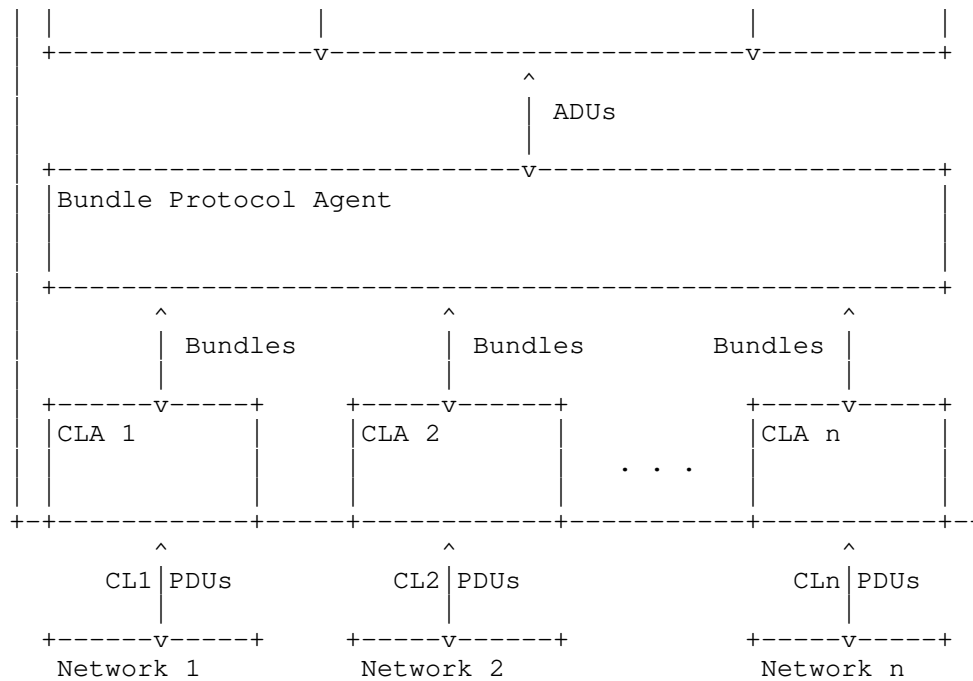


Figure 2: Components of a Bundle Node

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.

Convergence layer adapter - A convergence layer adapter (CLA) is a node component that sends and receives bundles on behalf of the BPA, utilizing the services of some 'native' protocol stack that is supported in one of the networks within which the node is functionally located.

Application agent - The application agent (AA) of a node is the node component that utilizes the BP services to effect communication for some user purpose. The application agent in turn has two elements, an administrative element and an application-specific element.

Application-specific element - The application-specific element of an AA is the node component that constructs, requests transmission of, accepts delivery of, and processes units of user application data.

Administrative element - The administrative element of an AA is the node component that constructs and requests transmission of administrative records (defined below), including status reports, and accepts delivery of and processes any administrative records that the node receives.

Administrative record - A BP administrative record is an application data unit that is exchanged between the administrative elements of nodes' application agents for some BP administrative purpose. The only administrative record defined in this specification is the status report, discussed later.

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 common identifier, called a "bundle endpoint ID" (or, in this document, simply "endpoint ID"; endpoint IDs are described in detail in Section 4.5.5.1 below.

Singleton endpoint - A singleton endpoint is an endpoint that always contains exactly one member.

Registration - A registration is the state machine characterizing a given node's membership in a given endpoint. Any single registration has an associated delivery failure action as defined below and must at any time be in one of two states: Active or Passive. Registrations are local; information about a node's registrations is not expected to be available at other nodes, and the Bundle Protocol does not include a mechanism for distributing information about registrations.

Delivery - 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 any relevant metadata (an implementation matter), has been presented to the node's application agent in a manner consistent with the state of that registration.

Deliverability - 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) the bundle has not yet been "abandoned" (as defined below) subject to this registration.

Abandonment - To abandon a bundle subject to some registration is to assert that the bundle is not deliverable subject to that registration.

Delivery failure action - The delivery failure action of a registration is the action that is to be taken when a bundle that is "deliverable" subject to that registration is received at a time when the registration is in the Passive state.

Destination - The destination of a bundle is the endpoint comprising the node(s) at which the bundle is to be delivered (as defined above).

Transmission - A transmission is an attempt by a node's BPA to cause copies of a bundle to be delivered to one or more of the nodes that are members of some endpoint (the bundle's destination) in response to a transmission request issued by the node's application agent.

Forwarding - To forward a bundle to a node is to invoke the services of one or more CLAs in a sustained effort to cause a copy of the bundle to be received by that node.

Discarding - To discard a bundle is to cease all operations on the bundle and functionally erase all references to it. The specific procedures by which this is accomplished are an implementation matter.

Retention constraint - A retention constraint is an element of the state of a bundle that prevents the bundle from being discarded. That is, a bundle cannot be discarded while it has any retention constraints.

Deletion - To delete a bundle is to remove unconditionally all of the bundle's retention constraints, enabling the bundle to be discarded.

3.2. Discussion of BP concepts

Multiple instances of the same bundle (the same unit of DTN protocol data) might exist concurrently in different parts of a network -- possibly differing in some blocks -- 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 (copy), in that node's local memory, of some bundle that is in the network.

The 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 payload for a bundle forwarded in response to reception of a bundle is the payload of the received bundle.

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.

The manner in which the functions of the BPA are performed 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.

Every CLA implements its own thin layer of protocol, interposed between BP and the (usually "top") protocol(s) of the underlying native protocol stack; this "CL protocol" may only serve to multiplex and de-multiplex bundles to and from the underlying native protocol, or it may offer additional CL-specific functionality. The manner in which a CLA sends and receives bundles, as well as the definitions of CLAs and CL protocols, are beyond the scope of this specification.

Note that the administrative element of a node's application agent may itself, in some cases, function as a convergence-layer adapter. That is, outgoing bundles may be "tunneled" through encapsulating bundles:

- . An outgoing bundle constitutes a byte array. This byte array may, like any other, be presented to the bundle protocol agent as an application data unit that is to be transmitted to some endpoint.
- . The original bundle thus forms the payload of an encapsulating bundle that is forwarded using some other convergence-layer protocol(s).
- . When the encapsulating bundle is received, its payload is delivered to the peer application agent administrative element, which then instructs the bundle protocol agent to dispatch that original bundle in the usual way.

The purposes for which this technique may be useful (such as cross-domain security) are beyond the scope of this specification.

The only interface between the BPA and the application-specific element of the AA is the BP service interface. But between the BPA and the administrative element of the AA there is a (conceptual)

private control interface in addition to the BP service interface. This private control interface enables the BPA and the administrative element of the AA to direct each other to take action under specific circumstances.

In the case of a node that serves simply as a BP "router", 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.

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 are all registered in a single common endpoint and will all receive any data delivered at that endpoint. *Note* too that any given bundle node might be registered in multiple bundle endpoints and receive all data delivered at each of those endpoints.

Recall that every node, by definition, includes an application agent which in turn includes an administrative element, which exchanges administrative records with the administrative elements of other nodes. As such, every node is permanently, structurally registered in the singleton endpoint at which administrative records received from other nodes are delivered. Registration in no other endpoint can ever be assumed to be permanent. This endpoint, termed the node's "administrative endpoint", is therefore uniquely and permanently associated with the node, and for this reason the ID of a node's administrative endpoint additionally serves as the "node ID" (see 4.1.5.2 below) of the node.

The destination of every bundle is an endpoint, which may or may not be singleton. The source of every bundle is a node, identified by node ID. Note, though, that the source node ID asserted in a given bundle may be the null endpoint ID (as described later) rather than the ID of the source node; bundles for which the asserted source node ID is the null endpoint ID are termed "anonymous" bundles.

Any number of transmissions may be concurrently undertaken by the bundle protocol agent of a given node.

When the bundle protocol agent of a node determines that a bundle must be forwarded to a node (either to a node that is a member of the bundle's destination endpoint or to some intermediate forwarding node) in the course of completing the successful transmission of

that bundle, the bundle protocol agent invokes the services of one or more CLAs in a sustained effort to cause a copy of the bundle to be received by that node.

Upon reception, the processing of a bundle that has been received by 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 newly 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.

The bundle protocol does not natively ensure delivery of a bundle to its destination. Data loss along the path to the destination node can be minimized by utilizing reliable convergence-layer protocols between neighbors on all segments of the end-to-end path, but for end-to-end bundle delivery assurance it will be necessary to develop extensions to the bundle protocol and/or application-layer mechanisms.

The bundle protocol is designed for extensibility. Bundle protocol extensions, documented elsewhere, may extend this specification by:

- . defining additional blocks;
- . defining additional administrative records;
- . defining additional bundle processing flags;
- . defining additional block processing flags;
- . defining additional types of bundle status reports;
- . defining additional bundle status report reason codes;
- . defining additional mandates and constraints on processing that conformant bundle protocol agents must perform at specified points in the inbound and outbound bundle processing cycles.

3.3. Services Offered by Bundle Protocol Agents

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

- . commencing a registration (registering the node in an endpoint);
- . terminating a registration;
- . switching a registration between Active and Passive states;
- . transmitting a bundle to an identified bundle endpoint;
- . canceling a transmission;

- . polling a registration that is in the Passive state;
- . delivering a received bundle.

Note that the details of registration functionality are an implementation matter and are beyond the scope of this specification.

4. Bundle Format

4.1. Bundle Structure

The format of bundles SHALL conform to the Concise Binary Object Representation (CBOR [RFC8949]).

Cryptographic verification of a block is possible only if the sequence of octets on which the verifying node computes its hash - the canonicalized representation of the block - is identical to the sequence of octets on which the hash declared for that block was computed. To ensure that blocks are always in canonical representation when they are transmitted and received, the CBOR representations of the values of all fields in all blocks must conform to the rules for Canonical CBOR as specified in [RFC8949].

Each bundle SHALL be a concatenated sequence of at least two blocks, represented as a CBOR indefinite-length array. The first block in the sequence (the first item of the array) MUST be a primary bundle block in CBOR representation as described below; the bundle MUST have exactly one primary bundle block. The primary block MUST be followed by one or more canonical bundle blocks (additional array items) in CBOR representation as described in 4.3.2 below. Every block following the primary block SHALL be the CBOR representation of a canonical block. The last such block MUST be a payload block; the bundle MUST have exactly one payload block. The payload block SHALL be followed by a CBOR "break" stop code, terminating the array.

(Note that, while CBOR permits considerable flexibility in the encoding of bundles, this flexibility must not be interpreted as inviting increased complexity in protocol data unit structure.)

Associated with each block of a bundle is a block number. The block number uniquely identifies the block within the bundle, enabling blocks (notably bundle security protocol blocks) to reference other blocks in the same bundle without ambiguity. The block number of the primary block is implicitly zero; the block numbers of all other blocks are explicitly stated in block headers as noted below. Block

numbering is unrelated to the order in which blocks are sequenced in the bundle. The block number of the payload block is always 1.

An implementation of the Bundle Protocol MAY discard any sequence of bytes that does not conform to the Bundle Protocol specification.

An implementation of the Bundle Protocol MAY accept a sequence of bytes that does not conform to the Bundle Protocol specification (e.g., one that represents data elements in fixed-length arrays rather than indefinite-length arrays) and transform it into conformant BP structure before processing it. Procedures for accomplishing such a transformation are beyond the scope of this specification.

4.2. BP Fundamental Data Structures

4.2.1. CRC Type

CRC type is an unsigned integer type code for which the following values (and no others) are valid:

- . 0 indicates "no CRC is present."
- . 1 indicates "a standard X-25 CRC-16 is present." [CRC16]
- . 2 indicates "a standard CRC32C (Castagnoli) CRC-32 is present." [RFC4960]

CRC type SHALL be represented as a CBOR unsigned integer.

For examples of CRC32C CRCs, see Appendix A.4 of [RFC7143].

Note that more robust protection of BP data integrity, as needed, may be provided by means of Block Integrity Blocks as defined in the Bundle Security Protocol [BPSEC]).

4.2.2. CRC

CRC SHALL be omitted from a block if and only if the block's CRC type code is zero.

When not omitted, the CRC SHALL be represented as a CBOR byte string of two bytes (that is, CBOR additional information 2, if CRC type is 1) or of four bytes (that is, CBOR additional information 4, if CRC type is 2); in each case the sequence of bytes SHALL constitute an unsigned integer value (of 16 or 32 bits, respectively) in network byte order.

4.2.3. Bundle Processing Control Flags

Bundle processing control flags assert properties of the bundle as a whole rather than of any particular block of the bundle. They are conveyed in the primary block of the bundle.

The following properties are asserted by the bundle processing control flags:

- . The bundle is a fragment. (Boolean)
- . The bundle's payload is an administrative record. (Boolean)
- . The bundle must not be fragmented. (Boolean)
- . Acknowledgment by the user application is requested. (Boolean)
- . Status time is requested in all status reports. (Boolean)
- . Flags requesting types of status reports (all Boolean):
 - o Request reporting of bundle reception.
 - o Request reporting of bundle forwarding.
 - o Request reporting of bundle delivery.
 - o Request reporting of bundle deletion.

If the bundle processing control flags indicate that the bundle's application data unit is an administrative record, then all status report request flag values MUST be zero.

If the bundle's source node is omitted (i.e., the source node ID is the ID of the null endpoint, which has no members as discussed below; this option enables anonymous bundle transmission), then the bundle is not uniquely identifiable and all bundle protocol features that rely on bundle identity must therefore be disabled: the "Bundle must not be fragmented" flag value MUST be 1 and all status report request flag values MUST be zero.

Bundle processing control flags that are unrecognized MUST be ignored, as future definitions of additional flags might not be integrated simultaneously into the Bundle Protocol implementations operating at all nodes.

The bundle processing control flags SHALL be represented as a CBOR unsigned integer item, the value of which SHALL be processed as a bit field indicating the control flag values as follows (note that bit numbering in this instance is reversed from the usual practice, beginning with the low-order bit instead of the high-order bit, in recognition of the potential definition of additional control flag values in the future):

- . Bit 0 (the low-order bit, 0x000001): bundle is a fragment.
- . Bit 1 (0x000002): payload is an administrative record.
- . Bit 2 (0x000004): bundle must not be fragmented.
- . Bit 3 (0x000008): reserved.
- . Bit 4 (0x000010): reserved.
- . Bit 5 (0x000020): user application acknowledgement is requested.
- . Bit 6 (0x000040): status time is requested in all status reports.
- . Bit 7 (0x000080): reserved.
- . Bit 8 (0x000100): reserved.
- . Bit 9 (0x000200): reserved.
- . Bit 10 (0x000400): reserved.
- . Bit 11 (0x000800): reserved.
- . Bit 12 (0x001000): reserved.
- . Bit 13 (0x002000): reserved.
- . Bit 14 (0x004000): bundle reception status reports are requested.
- . Bit 15 (0x008000): reserved.
- . Bit 16 (0x010000): bundle forwarding status reports are requested.
- . Bit 17 (0x020000): bundle delivery status reports are requested.
- . Bit 18 (0x040000): bundle deletion status reports are requested.
- . Bits 19-20 are reserved.
- . Bits 21-63 are unassigned.

4.2.4. Block Processing Control Flags

The block processing control flags assert properties of canonical bundle blocks. They are conveyed in the header of the block to which they pertain.

Block processing control flags that are unrecognized MUST be ignored, as future definitions of additional flags might not be integrated simultaneously into the Bundle Protocol implementations operating at all nodes.

The block processing control flags SHALL be represented as a CBOR unsigned integer item, the value of which SHALL be processed as a

bit field indicating the control flag values as follows (note that bit numbering in this instance is reversed from the usual practice, beginning with the low-order bit instead of the high-order bit, for agreement with the bit numbering of the bundle processing control flags):

- . Bit 0(the low-order bit, 0x01): block must be replicated in every fragment.
- . Bit 1(0x02): transmission of a status report is requested if block can't be processed.
- . Bit 2(0x04): bundle must be deleted if block can't be processed.
- . Bit 3(0x08): reserved.
- . Bit 4(0x10): block must be removed from bundle if it can't be processed.
- . Bit 5(0x20): reserved.
- . Bit 6 (0x40): reserved.
- . Bits 7-63 are unassigned.

For each bundle whose bundle processing control flags indicate that the bundle's application data unit is an administrative record, or whose source node ID is the null endpoint ID as defined below, the value of the "Transmit status report if block can't be processed" flag in every canonical block of the bundle MUST be zero.

4.2.5. Identifiers

4.2.5.1. Endpoint ID

The destinations of bundles are bundle endpoints, identified by text strings termed "endpoint IDs" (see Section 3.1). Each endpoint ID (EID) is 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" >

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. Each scheme that may be used to form a BP endpoint ID must be added to the registry of URI scheme code numbers for Bundle Protocol maintained by IANA as described in Section 10; association of a unique URI scheme code number with each scheme name in this registry helps to enable compact representation of endpoint IDs in bundle blocks. Note that the set of allowable schemes is effectively unlimited. Any scheme conforming to [URIREG] may be added to the URI scheme code number registry and thereupon used in a bundle protocol endpoint ID.

Each entry in the URI scheme code number registry MUST contain a reference to a scheme code number definition document, which defines the manner in which the scheme-specific part of any URI formed in that scheme is parsed and interpreted and MUST be encoded, in CBOR representation, for transmission as a BP endpoint ID. The scheme code number definition document may also contain information as to (a) which convergence-layer protocol(s) may be used to forward a bundle to a BP destination endpoint identified by such an ID, and (b) how the ID of the convergence-layer protocol endpoint to use for that purpose can be inferred from that destination endpoint ID.

Note that, although endpoint IDs are URIs, implementations of the BP service interface may support expression of endpoint IDs in some internationalized manner (e.g., Internationalized Resource Identifiers (IRIs); see [RFC3987]).

Each BP endpoint ID (EID) SHALL be represented as a CBOR array comprising two items.

The first item of the array SHALL be the code number identifying the endpoint ID's URI scheme, as defined in the registry of URI scheme code numbers for Bundle Protocol. Each URI scheme code number SHALL be represented as a CBOR unsigned integer.

The second item of the array SHALL be the applicable CBOR representation of the scheme-specific part (SSP) of the EID, defined as noted in the references(s) for the URI scheme code number registry entry for the EID's URI scheme.

4.2.5.1.1. The "dtn" URI scheme

The "dtn" scheme supports the identification of BP endpoints by arbitrarily expressive character strings. It is specified as follows:

Scheme syntax: This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234].

dtn-uri = "dtn:" ("none" / dtn-hier-part)

dtn-hier-part = "//" node-name name-delim demux ; a path-rootless

node-name = 1*(ALPHA/DIGIT/"-"/"."/"_") reg-name

name-delim = "/"

demux = *VCHAR

Scheme semantics: URIs of the dtn scheme are used as endpoint identifiers in the Delay-Tolerant Networking (DTN) Bundle Protocol (BP) as described in the present document.

The endpoint ID "dtn:none" identifies the "null endpoint", the endpoint that by definition never has any members.

All BP endpoints identified by all other dtn-scheme endpoint IDs for which the first character of demux is a character other than '~' (tilde) are singleton endpoints. All BP endpoints identified by dtn-scheme endpoint IDs for which the first character *is* '~' (tilde) are *not* singleton endpoints.

A dtn-scheme endpoint ID for which the demux is of length zero MAY identify the administrative endpoint for the node identified by node-name, and as such may serve as a node ID. No dtn-scheme endpoint ID for which the demux is of non-zero length may do so.

Note that these syntactic rules impose constraints on dtn-scheme endpoint IDs that were not imposed by the original specification of the dtn scheme as provided in [RFC5050]. It is believed that the dtn-scheme endpoint IDs employed by BP applications conforming to [RFC5050] are in most cases unlikely to be in violation of these rules, but the developers of such applications are advised of the potential for compromised interoperoperation.

Encoding considerations: For transmission as a BP endpoint ID, the scheme-specific part of a URI of the dtn scheme SHALL be represented as a CBOR text string unless the EID's SSP is "none", in which case the SSP SHALL be represented as a CBOR unsigned integer with the value zero. For all other purposes, URIs of the dtn scheme are encoded exclusively in US-ASCII characters.

Interoperability considerations: none.

Security considerations:

- . Reliability and consistency: none of the BP endpoints identified by the URIs of the dtn scheme are guaranteed to be reachable at any time, and the identity of the processing entities operating on those endpoints is never guaranteed by the Bundle Protocol itself. Bundle authentication as defined by the Bundle Security Protocol is required for this purpose.
- . Malicious construction: malicious construction of a conformant dtn-scheme URI is limited to the malicious selection of node names and the malicious selection of demux strings. That is, a maliciously constructed dtn-scheme URI could be used to direct

a bundle to an endpoint that might be damaged by the arrival of that bundle or, alternatively, to declare a false source for a bundle and thereby cause incorrect processing at a node that receives the bundle. In both cases (and indeed in all bundle processing), the node that receives a bundle should verify its authenticity and validity before operating on it in any way.

- . Back-end transcoding: the limited expressiveness of URIs of the dtn scheme effectively eliminates the possibility of threat due to errors in back-end transcoding.
- . Rare IP address formats: not relevant, as IP addresses do not appear anywhere in conformant dtn-scheme URIs.
- . Sensitive information: because dtn-scheme URIs are used only to represent the identities of Bundle Protocol endpoints, the risk of disclosure of sensitive information due to interception of these URIs is minimal. Examination of dtn-scheme URIs could be used to support traffic analysis; where traffic analysis is a plausible danger, bundles should be conveyed by secure convergence-layer protocols that do not expose endpoint IDs.
- . Semantic attacks: the simplicity of dtn-scheme URI syntax minimizes the possibility of misinterpretation of a URI by a human user.

4.2.5.1.2. The "ipn" URI scheme

The "ipn" scheme supports the identification of BP endpoints by pairs of unsigned integers, for compact representation in bundle blocks. It is specified as follows:

Scheme syntax: This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234], including the core ABNF syntax rule for DIGIT defined by that specification.

ipn-uri = "ipn:" ipn-hier-part

ipn-hier-part = node-nbr nbr-delim service-nbr ; a path-rootless

node-nbr = 1*DIGIT

nbr-delim = "."

service-nbr = 1*DIGIT

Scheme semantics: URIs of the ipn scheme are used as endpoint identifiers in the Delay-Tolerant Networking (DTN) Bundle Protocol (BP) as described in the present document.

All BP endpoints identified by ipn-scheme endpoint IDs are singleton endpoints.

An ipn-scheme endpoint ID for which service-nbr is zero MAY identify the administrative endpoint for the node identified by node-nbr, and as such may serve as a node ID. No ipn-scheme endpoint ID for which service-nbr is non-zero may do so.

Encoding considerations: For transmission as a BP endpoint ID, the scheme-specific part of a URI of the ipn scheme the SSP SHALL be represented as a CBOR array comprising two items. The first item of this array SHALL be the EID's node number (a number that identifies the node) represented as a CBOR unsigned integer. The second item of this array SHALL be the EID's service number (a number that identifies some application service) represented as a CBOR unsigned integer. For all other purposes, URIs of the ipn scheme are encoded exclusively in US-ASCII characters.

Interoperability considerations: none.

Security considerations:

- . Reliability and consistency: none of the BP endpoints identified by the URIs of the ipn scheme are guaranteed to be reachable at any time, and the identity of the processing entities operating on those endpoints is never guaranteed by the Bundle Protocol itself. Bundle authentication as defined by the Bundle Security Protocol [BPSEC] is required for this purpose.
- . Malicious construction: malicious construction of a conformant ipn-scheme URI is limited to the malicious selection of node numbers and the malicious selection of service numbers. That is, a maliciously constructed ipn-scheme URI could be used to direct a bundle to an endpoint that might be damaged by the arrival of that bundle or, alternatively, to declare a false source for a bundle and thereby cause incorrect processing at a node that receives the bundle. In both cases (and indeed in all bundle processing), the node that receives a bundle should verify its authenticity and validity before operating on it in any way.
- . Back-end transcoding: the limited expressiveness of URIs of the ipn scheme effectively eliminates the possibility of threat due to errors in back-end transcoding.
- . Rare IP address formats: not relevant, as IP addresses do not appear anywhere in conformant ipn-scheme URIs.
- . Sensitive information: because ipn-scheme URIs are used only to represent the identities of Bundle Protocol endpoints, the risk

- of disclosure of sensitive information due to interception of these URIs is minimal. Examination of ipn-scheme URIs could be used to support traffic analysis; where traffic analysis is a plausible danger, bundles should be conveyed by secure convergence-layer protocols that do not expose endpoint IDs.
- . Semantic attacks: the simplicity of ipn-scheme URI syntax minimizes the possibility of misinterpretation of a URI by a human user.

4.2.5.2. Node ID

For many purposes of the Bundle Protocol it is important to identify the node that is operative in some context.

As discussed in 3.1 above, nodes are distinct from endpoints; specifically, an endpoint is a set of zero or more nodes. But rather than define a separate namespace for node identifiers, we instead use endpoint identifiers to identify nodes as discussed in 3.2 above. Formally:

- . Every node is, by definition, permanently registered in the singleton endpoint at which administrative records are delivered to its application agent's administrative element, termed the node's "administrative endpoint".
- . As such, the EID of a node's administrative endpoint SHALL uniquely identify that node.
- . A "node ID" is an EID that identifies the administrative endpoint of a node.

4.2.6. DTN Time

A DTN time is an unsigned integer indicating the number of milliseconds that have elapsed since the DTN Epoch, 2000-01-01 00:00:00 +0000 (UTC). DTN time is not affected by leap seconds.

Each DTN time SHALL be represented as a CBOR unsigned integer item. Implementers need to be aware that DTN time values conveyed in CBOR representation in bundles will nearly always exceed $(2^{32} - 1)$; the manner in which a DTN time value is represented in memory is an implementation matter. The DTN time value zero indicates that the time is unknown.

4.2.7. Creation Timestamp

Each bundle's creation timestamp SHALL be represented as a CBOR array comprising two items.

The first item of the array, termed "bundle creation time", SHALL be the DTN time at which the transmission request was received that resulted in the creation of the bundle, represented as a CBOR unsigned integer.

The second item of the array, termed the creation timestamp's "sequence number", SHALL be the latest value (as of the time at which the transmission request was received) of a monotonically increasing positive integer counter managed by the source node's bundle protocol agent, represented as a CBOR unsigned integer. The sequence counter MAY be reset to zero whenever the current time advances by one millisecond.

For nodes that lack accurate clocks, it is recommended that bundle creation time be set to zero and that the counter used as the source of the bundle sequence count never be reset to zero.

Note that, in general, the creation of two distinct bundles with the same source node ID and bundle creation timestamp may result in unexpected network behavior and/or suboptimal performance. The combination of source node ID and bundle creation timestamp serves to identify a single transmission request, enabling it to be acknowledged by the receiving application (provided the source node ID is not the null endpoint ID).

4.2.8. Block-type-specific Data

Block-type-specific data in each block (other than the primary block) SHALL be the applicable CBOR representation of the content of the block. Details of this representation are included in the specification defining the block type.

4.3. Block Structures

This section describes the primary block in detail and non-primary blocks in general. 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 [BPSEC]) may require that BP implementations conforming to those protocols construct and process additional blocks.

4.3.1. Primary Bundle Block

The primary bundle block contains the basic information needed to forward bundles to their destinations.

Each primary block SHALL be represented as a CBOR array; the number of elements in the array SHALL be 8 (if the bundle is not a fragment and the block has no CRC), 9 (if the block has a CRC and the bundle is not a fragment), 10 (if the bundle is a fragment and the block has no CRC), or 11 (if the bundle is a fragment and the block has a CRC).

The primary block of each bundle SHALL be immutable. The CBOR-encoded values of all fields in the primary block MUST remain unchanged from the time the block is created to the time it is delivered.

The fields of the primary bundle block SHALL be as follows, listed in the order in which they MUST appear:

Version: An unsigned integer value indicating the version of the bundle protocol that constructed this block. The present document describes version 7 of the bundle protocol. Version number SHALL be represented as a CBOR unsigned integer item.

Bundle Processing Control Flags: The Bundle Processing Control Flags are discussed in Section 4.2.3. above.

CRC Type: CRC Type codes are discussed in Section 4.2.1. above. The CRC Type code for the primary block MAY be zero if the bundle contains a BPsec [BPSEC] Block Integrity Block whose target is the primary block; otherwise the CRC Type code for the primary block MUST be non-zero.

Destination EID: The Destination EID field identifies the bundle endpoint that is the bundle's destination, i.e., the endpoint that contains the node(s) at which the bundle is to be delivered.

Source node ID: The Source node ID field identifies the bundle node at which the bundle was initially transmitted, except that Source node ID may be the null endpoint ID in the event that the bundle's source chooses to remain anonymous.

Report-to EID: The Report-to EID field identifies the bundle endpoint to which status reports pertaining to the forwarding and delivery of this bundle are to be transmitted.

Creation Timestamp: The creation timestamp comprises two unsigned integers that, together with the source node ID and (if the bundle is a fragment) the fragment offset and payload length, serve to identify the bundle. See 4.2.7 above for the definition of this field.

Lifetime: The lifetime field is an unsigned integer that indicates the time at which the bundle's payload will no longer be useful, encoded as a number of milliseconds past the creation time. (For high-rate deployments with very brief disruptions, fine-grained expression of bundle lifetime may be useful.) When a bundle's age exceeds its lifetime, bundle nodes need no longer retain or forward the bundle; the bundle SHOULD be deleted from the network.

If the asserted lifetime for a received bundle is so lengthy that retention of the bundle until its expiration time might degrade operation of the node at which the bundle is received, or if the bundle protocol agent of that node determines that the bundle must be deleted in order to prevent network performance degradation (e.g., the bundle appears to be part of a denial-of-service attack), then that bundle protocol agent MAY impose a temporary overriding lifetime of shorter duration; such overriding lifetime SHALL NOT replace the lifetime asserted in the bundle but SHALL serve as the bundle's effective lifetime while the bundle resides at that node. Procedures for imposing lifetime overrides are beyond the scope of this specification.

For bundles originating at nodes that lack accurate clocks, it is recommended that bundle age be obtained from the Bundle Age extension block (see 4.4.2 below) rather than from the difference between current time and bundle creation time. Bundle lifetime SHALL be represented as a CBOR unsigned integer item.

Fragment offset: If and only if the Bundle Processing Control Flags of this Primary block indicate that the bundle is a fragment, fragment offset SHALL be present in the primary block. Fragment offset SHALL be represented as a CBOR unsigned integer indicating the offset from the start of the original application data unit at which the bytes comprising the payload of this bundle were located.

Total Application Data Unit Length: If and only if the Bundle Processing Control Flags of this Primary block indicate that the bundle is a fragment, total application data unit length SHALL be present in the primary block. Total application data unit length SHALL be represented as a CBOR unsigned integer indicating the total length of the original application data unit of which this bundle's payload is a part.

CRC: A CRC SHALL be present in the primary block unless the bundle includes a BPsec [BPSEC] Block Integrity Block whose target is the primary block, in which case a CRC MAY be present in the primary block. The length and nature of the CRC SHALL be as indicated by the CRC type. The CRC SHALL be computed over the concatenation of

all bytes (including CBOR "break" characters) of the primary block including the CRC field itself, which for this purpose SHALL be temporarily populated with all bytes set to zero.

4.3.2. Canonical Bundle Block Format

Every block other than the primary block (all such blocks are termed "canonical" blocks) SHALL be represented as a CBOR array; the number of elements in the array SHALL be 5 (if CRC type is zero) or 6 (otherwise).

The fields of every canonical block SHALL be as follows, listed in the order in which they MUST appear:

- . Block type code, an unsigned integer. Bundle block type code 1 indicates that the block is a bundle payload block. Block type codes 2 through 9 are explicitly reserved as noted later in this specification. Block type codes 192 through 255 are not reserved and are available for private and/or experimental use. All other block type code values are reserved for future use.
- . Block number, an unsigned integer as discussed in 4.1 above. Block number SHALL be represented as a CBOR unsigned integer.
- . Block processing control flags as discussed in Section 4.2.4 above.
- . CRC type as discussed in Section 4.2.1 above.
- . Block-type-specific data represented as a single definite-length CBOR byte string, i.e., a CBOR byte string that is not of indefinite length. For each type of block, the block-type-specific data byte string is the serialization, in a block-type-specific manner, of the data conveyed by that type of block; definitions of blocks are required to define the manner in which block-type-specific data are serialized within the block-type-specific data field. For the Payload Block in particular (block type 1), the block-type-specific data field, termed the "payload", SHALL be an application data unit, or some contiguous extent thereof, represented as a definite-length CBOR byte string.
- . If and only if the value of the CRC type field of this block is non-zero, a CRC. If present, the length and nature of the CRC SHALL be as indicated by the CRC type and the CRC SHALL be computed over the concatenation of all bytes of the block (including CBOR "break" characters) including the CRC field itself, which for this purpose SHALL be temporarily populated with all bytes set to zero.

4.4. Extension Blocks

"Extension blocks" are all blocks other than the primary and payload blocks. Three types of extension blocks are defined below. All implementations of the Bundle Protocol specification (the present document) MUST include procedures for recognizing, parsing, and acting on, but not necessarily producing, these types of extension blocks.

The specifications for additional types of extension blocks must indicate whether or not BP implementations conforming to those specifications must recognize, parse, act on, and/or produce blocks of those types. As not all nodes will necessarily instantiate BP implementations that conform to those additional specifications, it is possible for a node to receive a bundle that includes extension blocks that the node cannot process. The values of the block processing control flags indicate the action to be taken by the bundle protocol agent when this is the case.

No mandated procedure in this specification is unconditionally dependent on the absence or presence of any extension block. Therefore any bundle protocol agent MAY insert or remove any extension block in any bundle, subject to all mandates in the Bundle Protocol specification and all extension block specifications to which the node's BP implementation conforms. Note that removal of an extension block will probably disable one or more elements of bundle processing that were intended by the BPA that inserted that block. In particular, note that removal of an extension block that is one of the targets of a BPsec security block may render the bundle unverifiable.

The following extension blocks are defined in the current document.

4.4.1. Previous Node

The Previous Node block, block type 6, identifies the node that forwarded this bundle to the local node (i.e., to the node at which the bundle currently resides); its block-type-specific data is the node ID of that forwarder node which SHALL take the form of a node ID represented as described in Section 4.2.5.2. above. If the local node is the source of the bundle, then the bundle MUST NOT contain any Previous Node block. Otherwise the bundle SHOULD contain one (1) occurrence of this type of block and MUST NOT contain more than one.

4.4.2. Bundle Age

The Bundle Age block, block type 7, contains the number of milliseconds that have elapsed between the time the bundle was created and time at which it was most recently forwarded. It is intended for use by nodes lacking access to an accurate clock, to aid in determining the time at which a bundle's lifetime expires. The block-type-specific data of this block is an unsigned integer containing the age of the bundle in milliseconds, which SHALL be represented as a CBOR unsigned integer item. (The age of the bundle is the sum of all known intervals of the bundle's residence at forwarding nodes, up to the time at which the bundle was most recently forwarded, plus the summation of signal propagation time over all episodes of transmission between forwarding nodes. Determination of these values is an implementation matter.) If the bundle's creation time is zero, then the bundle MUST contain exactly one (1) occurrence of this type of block; otherwise, the bundle MAY contain at most one (1) occurrence of this type of block. A bundle MUST NOT contain multiple occurrences of the bundle age block, as this could result in processing anomalies.

4.4.3. Hop Count

The Hop Count block, block type 10, contains two unsigned integers, hop limit and hop count. A "hop" is here defined as an occasion on which a bundle was forwarded from one node to another node. Hop limit MUST be in the range 1 through 255. The hop limit value SHOULD NOT be changed at any time after creation of the Hop Count block; the hop count value SHOULD initially be zero and SHOULD be increased by 1 on each hop.

The hop count block is mainly intended as a safety mechanism, a means of identifying bundles for removal from the network that can never be delivered due to a persistent forwarding error. Hop count is particularly valuable as a defense against routing anomalies that might cause a bundle to be forwarded in a cyclical "ping-pong" fashion between two nodes. When a bundle's hop count exceeds its hop limit, the bundle SHOULD be deleted for the reason "hop limit exceeded", following the bundle deletion procedure defined in Section 5.10.

Procedures for determining the appropriate hop limit for a bundle are beyond the scope of this specification.

The block-type-specific data in a hop count block SHALL be represented as a CBOR array comprising two items. The first item of this array SHALL be the bundle's hop limit, represented as a CBOR

unsigned integer. The second item of this array SHALL be the bundle's hop count, represented as a CBOR unsigned integer. A bundle MAY contain one occurrence of this type of block but MUST NOT contain more than one.

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. Supplementary DTN protocol specifications (including, but not restricted to, the Bundle Security Protocol [BPSEC]) may augment, override, or supersede the mandates of this document.

5.1. Generation of Administrative Records

All transmission of bundles is in response to bundle transmission requests presented by nodes' application agents. When required to "generate" an administrative record (such as a bundle status report), 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.

Status reports are relatively small bundles. Moreover, even when the generation of status reports is enabled the decision on whether or not to generate a requested status report is left to the discretion of the bundle protocol agent. Nonetheless, note that requesting status reports for any single bundle might easily result in the generation of $(1 + (2 * (N-1)))$ status report bundles, where N is the number of nodes on the path from the bundle's source to its destination, inclusive. That is, the requesting of status reports for large numbers of bundles could result in an unacceptable increase in the bundle traffic in the network. For this reason, the generation of status reports MUST be disabled by default and enabled only when the risk of excessive network traffic is deemed acceptable. Mechanisms that could assist in assessing and mitigating this risk, 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 "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.

5.2. Bundle Transmission

The steps in processing a bundle transmission request are:

Step 1: Transmission of the bundle is initiated. An outbound bundle MUST be created per the parameters of the bundle transmission request, with the retention constraint "Dispatch pending". The source node ID of the bundle MUST be either the null endpoint ID, indicating that the source of the bundle is anonymous, or else the EID of a singleton endpoint whose only member is the node of which the BPA is a component.

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

5.3. Bundle Dispatching

(Note that this procedure is initiated only following completion of Step 4 of Section 5.6.)

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 and for the purposes of all subsequent processing of this bundle at this node the node's membership in the bundle's destination endpoint SHALL be disavowed; specifically, even though the node is a member of the bundle's destination endpoint, the node SHALL NOT undertake to forward the bundle to itself in the course of performing the procedure described in Section 5.4.

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 (that is, rendered inadvisable) for any of the reasons listed in the IANA registry of Bundle Status Report Reason Codes (see section 10.5 below), whose initial contents are listed in Figure 4. In particular:

- . The bundle protocol agent MAY choose either to forward the bundle directly to its destination node(s) (if possible) or to forward the bundle to some other node(s) for further forwarding. The manner in which this decision is made may depend on the scheme name in the destination endpoint ID and/or on other state but in any case is beyond the scope of this document; one possible mechanism is described in [SABR]. If the BPA elects to forward the bundle to some other node(s) for further forwarding but finds it impossible to select any node(s) to forward the bundle to, then forwarding is contraindicated.
- . Provided the bundle protocol agent succeeded in selecting the node(s) to forward the bundle to, the bundle protocol agent MUST subsequently select the convergence layer adapter(s) whose services will enable the node to send the bundle to those nodes. The manner in which specific appropriate convergence layer adapters are selected is beyond the scope of this document; the TCP convergence-layer adapter [TCPCL] MUST be implemented when some or all of the bundles forwarded by the bundle protocol agent must be forwarded via the Internet but may not be appropriate for the forwarding of any particular bundle. If the agent finds it impossible to select any appropriate convergence layer adapter(s) 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 the IANA registry of Bundle Status Report Reason Codes (see section 10.5 below), then the Forwarding Contraindicated procedure defined in Section 5.4.1 MUST be followed; the remaining steps of Section 5.4 are skipped at this time.

Step 4: For each node 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 that node. Determining the time at which the bundle protocol agent invokes convergence layer adapter services is a BPA implementation

matter. Determining the time at which each convergence layer adapter subsequently responds to this service invocation by sending the bundle is a convergence-layer adapter implementation matter. Note that:

- . If the bundle has a Previous Node block, as defined in 4.4.1 above, then that block **MUST** be removed from the bundle before the bundle is forwarded.
- . If the bundle protocol agent is configured to attach Previous Node blocks to forwarded bundles, then a Previous Node block containing the node ID of the forwarding node **MUST** be inserted into the bundle before the bundle is forwarded.
- . If the bundle has a bundle age block, as defined in 4.4.2. above, then at the last possible moment before the CLA initiates conveyance of the bundle via the CL protocol the bundle age value **MUST** be increased by the difference between the current time and the time at which the bundle was received (or, if the local node is the source of the bundle, created).

Step 5: 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, processing may depend on the results of those procedures.

If completion of the data sending procedures by all selected convergence layer adapters has not resulted in successful forwarding of the bundle (an implementation-specific determination that is beyond the scope of this specification), then the bundle protocol agent **MAY** choose (in an implementation-specific manner, again beyond the scope of this specification) to initiate another attempt to forward the bundle. In that event, processing proceeds from Step 4. The minimum number of times a given node will initiate another forwarding attempt for any single bundle in this event (a number which may be zero) is a node configuration parameter that must be exposed to other nodes in the network to the extent that this is required by the operating environment.

If completion of the data sending procedures by all selected convergence layer adapters **HAS** resulted in successful forwarding of the bundle, or if it has not but the bundle protocol agent does not choose to initiate another attempt to forward the bundle, then:

- . If the "request reporting of bundle forwarding" flag in the bundle's status report request field is set to 1, and status reporting is enabled, then a bundle forwarding status report **SHOULD** be generated, destined for the bundle's report-to

- endpoint ID. The reason code on this bundle forwarding status report MUST be "no additional information".
- . If any applicable bundle protocol extensions mandate generation of status reports upon conclusion of convergence-layer data sending procedures, all such status reports SHOULD be generated with extension-mandated reason codes.
- . The bundle's "Forward pending" retention constraint MUST be removed.

5.4.1. Forwarding Contraindicated

The steps in responding to contraindication of forwarding are:

Step 1: The bundle protocol agent MUST determine whether or not to declare failure in forwarding the bundle. 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 Section 5.4.2 MUST be followed.

Otherwise, when - at some future time - the forwarding of this bundle ceases to be contraindicated, processing proceeds from Step 4 of Section 5.4.

5.4.2. Forwarding Failed

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

Step 1: The bundle protocol agent MAY forward the bundle back to the node that sent it, as identified by the Previous Node block, if present. This forwarding, if performed, SHALL be accomplished by performing Step 4 and Step 5 of section 5.4 where the sole node selected for forwarding SHALL be the node that sent the bundle.

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 Section 5.10 MUST be followed, citing the reason for which forwarding was determined to be contraindicated.

5.5. Bundle Expiration

A bundle expires when the bundle's age exceeds its lifetime as specified in the primary bundle block or as overridden by the bundle protocol agent. Bundle age MAY be determined by subtracting the

bundle's creation timestamp time from the current time if (a) that timestamp time is not zero and (b) the local node's clock is known to be accurate; otherwise bundle age MUST be obtained from the Bundle Age extension 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" (when the expired lifetime is the lifetime as specified in the primary block) or "traffic pared" (when the expired lifetime is a lifetime override as imposed by the bundle protocol agent): the bundle deletion procedure defined in Section 5.10 MUST be followed.

5.6. Bundle Reception

The steps in processing a bundle that has been 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, and status reporting is enabled, 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: CRCs SHOULD be computed for every block of the bundle that has an attached CRC. If any block of the bundle is malformed according to this specification (including syntactically invalid CBOR), or if any block has an attached CRC and the CRC computed for this block upon reception differs from that attached CRC, then the bundle protocol agent MUST delete the bundle for the reason "Block unintelligible". The bundle deletion procedure defined in Section 5.10 MUST be followed and all remaining steps of the bundle reception procedure MUST be skipped.

Step 4: 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, and status reporting is enabled, then a bundle reception status report with reason code "Block unsupported" 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

unsupported"; the bundle deletion procedure defined in Section 5.10 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 that the block must be discarded, then processing continues with the next extension block that the bundle protocol agent cannot process, if any; otherwise, processing proceeds from step 5.

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 the fragmentary bundle whose payload has been replaced by the reassembled application data unit (whether this bundle or a previously received fragment) proceeds from Step 2; otherwise, the retention constraint "Reassembly pending" MUST be added to the bundle and all remaining steps of this procedure MUST be skipped.

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

- . An additional implementation-specific delivery deferral procedure MAY optionally be associated with the registration.
- . If the registration is in the Active state, then the bundle MUST be delivered automatically as soon as it is the next bundle that is due for delivery according to the BPA's bundle delivery scheduling policy, an implementation matter.
- . If the registration is in the Passive state, or if delivery of the bundle fails for some implementation-specific reason, then the registration's delivery failure action MUST be taken. Delivery failure action MUST be one of the following:
 - o defer delivery 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 the Active state, and also perform any additional delivery deferral procedure associated with the registration; or

- o abandon delivery of the bundle subject to this registration (as defined in 3.1.).

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 and bundle status reporting is enabled, 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.

5.8. Bundle Fragmentation

It may at times be advantageous for bundle protocol agents to reduce the sizes of bundles in order to forward them. This might be the case, for example, if a node 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 node ID and creation timestamp as the original bundle - whose payloads MUST be the first N and the last $(M - N)$ bytes of the original bundle's payload, where $0 < N < M$.

Note that fragments are bundles and therefore may themselves be fragmented, so multiple episodes of 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. NOTE, however, that some combinations of bundle fragmentation, replication, and routing might result in unexpected traffic patterns.

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

fragmented bundle (that is, the bundle that is being fragmented). Note that the payloads of fragments resulting from different fragmentation episodes, in different parts of the network, may be overlapping subsets of the fragmented bundle's payload.

- . The primary block of each fragment MUST differ from that of the fragmented bundle, in that the bundle processing flags of the fragment MUST indicate that the bundle is a fragment and both fragment offset and total application data unit length must be provided. Additionally, the CRC of the primary block of the fragmented bundle, if any, MUST be replaced in each fragment by a new CRC computed for the primary block of that fragment.
- . The payload blocks of fragments will differ from that of the fragmented bundle as noted above.
- . If the fragmented bundle is not a fragment or is the fragment with offset zero, then all extension blocks of the fragmented bundle MUST be replicated in the fragment whose offset is zero.
- . Each of the fragmented bundle's extension blocks whose "Block must be replicated in every fragment" flag is set to 1 MUST be replicated in every fragment.
- . Beyond these rules, rules for the replication of extension blocks in the fragments must be defined in the specifications for those extension block types.

5.9. Application Data Unit Reassembly

Note that the bundle fragmentation procedure described in 5.8 above may result in the replacement of a single original bundle with an arbitrarily large number of fragmentary bundles. In order to be delivered at a destination node, the original bundle's payload must be reassembled from the payloads of those fragments.

The "material extents" of a received fragment's payload are all continuous sequences of bytes in that payload that do not overlap with the material extents of the payloads of any previously received fragments with the same source node ID and creation timestamp. If the concatenation - as informed by fragment offsets and payload lengths - of the material extents of the payloads of this fragment and all previously received fragments with the same source node ID and creation timestamp as this fragment forms a continuous byte array whose length is equal to the total application data unit length noted in the fragment's primary block, then:

- . This byte array -- the reassembled application data unit -- MUST replace the payload of that fragment whose material extents include the extent at offset zero. Note that this will

enable delivery of the reconstituted original bundle as described in Step 1 of 5.7.

- . The "Reassembly pending" retention constraint MUST be removed from every other fragment with the same source node ID and creation timestamp as this fragment.

Note: reassembly of application data units from fragments occurs at the nodes that are members of destination endpoints as necessary; an application data unit MAY also be reassembled at some other node on the path to the destination.

5.10. Bundle Deletion

The steps in deleting a bundle are:

Step 1: If the "request reporting of bundle deletion" flag in the bundle's status report request field is set to 1, and if status reporting is enabled, 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.

5.11. Discarding a Bundle

As soon as a bundle has no remaining retention constraints it MAY be discarded, thereby releasing any persistent storage that may have been allocated to it.

5.12. 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 cancelled". For this purpose, the procedure defined in Section 5.10 MUST be followed.

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. One type of administrative record has been defined to date: bundle status reports. Note that additional types of administrative records may be defined by supplementary DTN protocol specification documents.

Every administrative record consists of:

- . Record type code (an unsigned integer for which valid values are as defined below).
- . Record content in type-specific format.

Valid administrative record type codes are defined as follows:

Value	Meaning
1	Bundle status report.
(other)	Reserved for future use.

Figure 3: Administrative Record Type Codes

Each BP administrative record SHALL be represented as a CBOR array comprising two items.

The first item of the array SHALL be a record type code, which SHALL be represented as a CBOR unsigned integer.

The second element of this array SHALL be the applicable CBOR representation of the content of the record. Details of the CBOR representation of administrative record type 1 are provided below. Details of the CBOR representation of other types of administrative record type are included in the specifications defining those records.

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, forwarding, final delivery, and deletion. They are transmitted to the Report-to endpoints of bundles.

Each bundle status report SHALL be represented as a CBOR array. The number of elements in the array SHALL be either 6 (if the subject bundle is a fragment) or 4 (otherwise).

The first item of the bundle status report array SHALL be bundle status information represented as a CBOR array of at least 4 elements. The first four items of the bundle status information array shall provide information on the following four status assertions, in this order:

- . Reporting node received bundle.
- . Reporting node forwarded the bundle.
- . Reporting node delivered the bundle.
- . Reporting node deleted the bundle.

Each item of the bundle status information array SHALL be a bundle status item represented as a CBOR array; the number of elements in each such array SHALL be either 2 (if the value of the first item of this bundle status item is 1 AND the "Report status time" flag was set to 1 in the bundle processing flags of the bundle whose status is being reported) or 1 (otherwise). The first item of the bundle status item array SHALL be a status indicator, a Boolean value indicating whether or not the corresponding bundle status is asserted, represented as a CBOR Boolean value. The second item of the bundle status item array, if present, SHALL indicate the time (as reported by the local system clock, an implementation matter) at which the indicated status was asserted for this bundle, represented as a DTN time as described in Section 4.2.6. above.

The second item of the bundle status report array SHALL be the bundle status report reason code explaining the value of the status indicator, represented as a CBOR unsigned integer. Valid status report reason codes are registered in the IANA Bundle Status Report Reason Codes registry in the Bundle Protocol Namespace (see 10.5 below). The initial contents of that registry are listed in Figure 4 below but 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 [BPSEC]) may define additional reason codes.

+-----+-----+-----+-----+-----+-----+					
Value		Meaning			
+=====+		+=====+			
0		No additional information.			

+-----+-----+		
1	Lifetime expired.	
+-----+-----+		
2	Forwarded over unidirectional link.	
+-----+-----+		
3	Transmission canceled.	
+-----+-----+		
4	Depleted storage.	
+-----+-----+		
5	Destination endpoint ID unavailable.	
+-----+-----+		
6	No known route to destination from here.	
+-----+-----+		
7	No timely contact with next node on route.	
+-----+-----+		
8	Block unintelligible.	
+-----+-----+		
9	Hop limit exceeded.	
+-----+-----+		
10	Traffic pared (e.g., status reports).	
+-----+-----+		
11	Block unsupported.	
+-----+-----+		
(other)	Reserved for future use.	

Figure 4: Status Report Reason Codes

The third item of the bundle status report array SHALL be the source node ID identifying the source of the bundle whose status is being reported, represented as described in Section 4.2.5.1.1. above.

The fourth item of the bundle status report array SHALL be the creation timestamp of the bundle whose status is being reported, represented as described in Section 4.2.7. above.

The fifth item of the bundle status report array SHALL be present if and only if the bundle whose status is being reported contained a fragment offset. If present, it SHALL be the subject bundle's fragment offset represented as a CBOR unsigned integer item.

The sixth item of the bundle status report array SHALL be present if and only if the bundle whose status is being reported contained a fragment offset. If present, it SHALL be the length of the subject bundle's payload represented as a CBOR unsigned integer item.

Note that the forwarding parameters (such as lifetime, applicable security measures, etc.) of the bundle whose status is being reported MAY be reflected in the parameters governing the forwarding of the bundle that conveys a status report, but this is an implementation matter. Bundle protocol deployment experience to date has not been sufficient to suggest any clear guidance on this topic.

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, the following procedure must be followed:

Step 1: The administrative record must be constructed. If the administrative record references a bundle and the referenced bundle is a fragment, the administrative record **MUST** contain the fragment offset and fragment length.

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

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 a bundle node that is reachable via the convergence layer protocol;
- . notifying the bundle protocol agent of the disposition of its data sending procedures with regard to a bundle, upon concluding those procedures;
- . delivering to the bundle protocol agent a bundle that was sent by a 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 [BPSEC]) may expect convergence layer adapters that serve BP implementations conforming to those protocols to provide additional services such as reporting on the transmission and/or reception progress of individual bundles (at completion and/or incrementally), retransmitting data that were lost in transit, discarding bundle-conveying data units that the convergence layer protocol determines are corrupt or inauthentic, or reporting on the integrity and/or authenticity of delivered bundles.

In addition, bundle protocol relies on the capabilities of protocols at the convergence layer to minimize congestion in the store-carry-forward overlay network. The potentially long round-trip times characterizing delay-tolerant networks are incompatible with end-to-end reactive congestion control mechanisms, so convergence-layer protocols MUST provide rate limiting or congestion control.

8. Implementation Status

[NOTE to the RFC Editor: please remove this section before publication, as well as the reference to RFC 7942.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

At the time of this writing, there are six known implementations of the current document.

The first known implementation is microPCN (<https://upcn.eu/>). According to the developers:

The Micro Planetary Communication Network (uPCN) is a free software project intended to offer an implementation of Delay-tolerant Networking protocols for POSIX operating systems (well, and for Linux) plus for the ARM Cortex STM32F4 microcontroller series. More precisely it currently provides an implementation of

- . the Bundle Protocol (BP, RFC 5050),
- . version 6 of the Bundle Protocol version 7 specification draft,
- . the DTN IP Neighbor Discovery (IPND) protocol, and
- . a routing approach optimized for message-ferry micro LEO satellites.

uPCN is written in C and is built upon the real-time operating system FreeRTOS. The source code of uPCN is released under the "BSD 3-Clause License".

The project depends on an execution environment offering link layer protocols such as AX.25. The source code uses the USB subsystem to interact with the environment.

The second known implementation is PyDTN, developed by X-works, s.r.o (<https://x-works.sk/>). The final third of the implementation was developed during the IETF 101 Hackathon. According to the developers, PyDTN implements bundle coding/decoding and neighbor discovery. PyDTN is written in Python and has been shown to be interoperable with uPCN.

The third known implementation is "Terra" (<https://github.com/RightMesh/Terra/>), a Java implementation developed in the context of terrestrial DTN. It includes an implementation of a "minimal TCP" convergence layer adapter.

The fourth and fifth known implementations are products of cooperating groups at two German universities:

- . An implementation written in Go, licensed under GPLv3, is focused on being easily extensible suitable for research. It is maintained at the University of Marburg and can be accessed from <https://github.com/dtn7/dtn7-go>.
- . An implementation written in Rust, licensed under the MIT/Apache license, is intended for environments with limited resources or demanding safety and/or performance requirements. It is maintained at the Technical University of Darmstadt and can be accessed at <https://github.com/dtn7/dtn7-rs/>.

The sixth known implementation is the "bpv7" module in version 4.0.0 of the Interplanetary Overlay Network (ION) software maintained at the Jet Propulsion Laboratory, California Institute of Technology, for the U.S. National Aeronautics and Space Administration (NASA).

9. Security Considerations

The bundle protocol security architecture and the available security services are specified in an accompanying document, the Bundle Security Protocol (BPsec) specification [BPSEC]. Whenever Bundle Protocol security services (as opposed to the security services provided by overlying application protocols or underlying convergence-layer protocols) are required, those services SHALL be provided by BPsec rather than by some other mechanism with the same or similar scope.

A Bundle Protocol Agent (BPA) which sources, cryptographically verifies, and/or accepts a bundle MUST implement support for BPsec. Use of BPsec for a particular Bundle Protocol session is optional.

The BPsec extensions to Bundle Protocol enable each block of a bundle (other than a BPsec extension block) to be individually authenticated by a signature block (Block Integrity Block, or BIB) and also enable each block of a bundle other than the primary block (and the BPsec extension blocks themselves) to be individually encrypted by a Block Confidentiality Block (BCB).

Because the security mechanisms are extension blocks that are themselves inserted into the bundle, the protections they afford apply while the bundle is at rest, awaiting transmission at the next forwarding opportunity, as well as in transit.

Additionally, convergence-layer protocols that ensure authenticity of communication between adjacent nodes in BP network topology SHOULD be used where available, to minimize the ability of unauthenticated nodes to introduce inauthentic traffic into the network. Convergence-layer protocols that ensure confidentiality of communication between adjacent nodes in BP network topology SHOULD also be used where available, to minimize exposure of the bundle's primary block and other clear-text blocks, thereby offering some defense against traffic analysis.

In order to provide authenticity and/or confidentiality of communication between BP nodes, the convergence-layer protocol requires as input the name(s) of the expected communication peer(s). These must be supplied by the convergence-layer adapter. Details of the means by which the CLA determines which CL endpoint name(s) must be provided to the CL protocol are out of scope for this specification. Note, though, that when the CL endpoint names are a function of BP endpoint IDs, the correctness and authenticity of that mapping will be vital to the overall security properties that the CL provides to the system.

Note that, while the primary block must remain in the clear for routing purposes, the Bundle Protocol could be protected against traffic analysis to some extent by using bundle-in-bundle encapsulation [BIBE] to tunnel bundles to a safe forward distribution point: the encapsulated bundle could form the payload of an encapsulating bundle, and that payload block could be encrypted by a BCB.

Note that the generation of bundle status reports is disabled by default because malicious initiation of bundle status reporting

could result in the transmission of extremely large numbers of bundles, effecting a denial of service attack. Imposing bundle lifetime overrides would constitute one defense against such an attack.

Note also that the reception of large numbers of fragmentary bundles with very long lifetimes could constitute a denial of service attack, occupying storage while pending reassembly that will never occur. Imposing bundle lifetime overrides would, again, constitute one defense against such an attack.

This protocol makes use of absolute timestamps for several purposes. Provisions are included for nodes without accurate clocks to retain most of the protocol functionality, but nodes that are unaware that their clock is inaccurate may exhibit unexpected behavior.

10. IANA Considerations

The Bundle Protocol includes fields requiring registries managed by IANA.

10.1. Bundle Block Types

The current Bundle Block Types registry in the Bundle Protocol Namespace is augmented by adding a column identifying the version of the Bundle protocol (Bundle Protocol Version) that applies to the new values. IANA is requested to add the following values, as described in section 4.3.1, to the Bundle Block Types registry. The current values in the Bundle Block Types registry should have the Bundle Protocol Version set to the value "6", as shown below.

Bundle	Value	Description	Reference
Protocol			
Version			
none	0	Reserved	[RFC6255]
6,7	1	Bundle Payload Block	[RFC5050]
			RFC-to-be

	6		2		Bundle Authentication Block		[RFC6257]	
	6		3		Payload Integrity Block		[RFC6257]	
	6		4		Payload Confidentiality		[RFC6257]	
					Block			
	6		5		Previous-Hop Insertion Block		[RFC6259]	
	7		6		Previous node (proximate		RFC-to-be	
					sender)			
	7		7		Bundle age (in milliseconds)		RFC-to-be	
	6		8		Metadata Extension Block		[RFC6258]	
	6		9		Extension Security Block		[RFC6257]	
	7		10		Hop count (#prior xmit		RFC-to-be	
					attempts)			
	7		11-191		Unassigned			
	6,7		192-255		Reserved for Private and/or		[RFC5050],	
					Experimental Use		RFC-to-be	
+-----+-----+-----+-----+-----+								

10.2. Primary Bundle Protocol Version

IANA is requested to add the following value to the Primary Bundle Protocol Version registry in the Bundle Protocol Namespace.

+-----+-----+-----+-----+			
	Value		Description Reference
+-----+-----+-----+-----+			
	7		Assigned RFC-to-be
+-----+-----+-----+-----+			

Values 8-255 (rather than 7-255) are now Unassigned.

10.3. Bundle Processing Control Flags

The current Bundle Processing Control Flags registry in the Bundle Protocol Namespace is augmented by adding a column identifying the version of the Bundle protocol (Bundle Protocol Version) that applies to the new values. IANA is requested to add the following values, as described in section 4.1.3, to the Bundle Processing Control Flags registry. The current values in the Bundle Processing Control Flags registry should have the Bundle Protocol Version set to the value 6 or "6, 7", as shown below.

Bundle Processing Control Flags Registry

+-----+-----+-----+			
Bundle	Bit	Description	Reference
Protocol	Position		
Version	(right		
	to left)		
+-----+-----+-----+			
6,7	0	Bundle is a fragment	[RFC5050],
			RFC-to-be
6,7	1	Application data unit is an	[RFC5050],
		administrative record	RFC-to-be
6,7	2	Bundle must not be fragmented	[RFC5050],
			RFC-to-be
6	3	Custody transfer is requested	[RFC5050]
6	4	Destination endpoint is singleton	[RFC5050]
6,7	5	Acknowledgement by application	[RFC5050],
		is requested	RFC-to-be

	7		6		Status time requested in reports		RFC-to-be	
	6		7		Class of service, priority		[RFC5050]	
	6		8		Class of service, priority		[RFC5050]	
	6		9		Class of service, reserved		[RFC5050]	
	6		10		Class of service, reserved		[RFC5050]	
	6		11		Class of service, reserved		[RFC5050]	
	6		12		Class of service, reserved		[RFC5050]	
	6		13		Class of service, reserved		[RFC5050]	
	6,7		14		Request reporting of bundle		[RFC5050],	
					reception		RFC-to-be	
	6		15		Request reporting of custody		[RFC5050]	
					acceptance			
	6,7		16		Request reporting of bundle		[RFC5050],	
					forwarding		RFC-to-be	
	6,7		17		Request reporting of bundle		[RFC5050],	
					delivery		RFC-to-be	
	6,7		18		Request reporting of bundle		[RFC5050],	
					deletion		RFC-to-be	
	6,7		19		Reserved		[RFC5050],	
							RFC-to-be	
	6,7		20		Reserved		[RFC5050],	
							RFC-to-be	
			21-63		Unassigned			

+-----+-----+-----+

10.4. Block Processing Control Flags

The current Block Processing Control Flags registry in the Bundle Protocol Namespace is augmented by adding a column identifying the version of the Bundle protocol (Bundle Protocol Version) that applies to the related BP version. The current values in the Block Processing Control Flags registry should have the Bundle Protocol Version set to the value 6 or "6, 7", as shown below.

Block Processing Control Flags Registry

+-----+-----+-----+			
Bundle	Bit	Description	Reference
Protocol	Position		
Version	(right		
	to left)		
+-----+-----+-----+			
6,7	0	Block must be replicated in	[RFC5050],
		every fragment	RFC-to-be
6,7	1	Transmit status report if block	[RFC5050],
		can't be processed	RFC-to-be
6,7	2	Delete bundle if block can't be	[RFC5050],
		processed	RFC-to-be
6	3	Last block	[RFC5050]
6,7	4	Discard block if it can't be	[RFC5050],
		processed	RFC-to-be
6	5	Block was forwarded without	[RFC5050]
		being processed	
6	6	Block contains an EID reference	[RFC5050]

		field		
		7-63 Unassigned		
+-----+-----+-----+				

10.5. Bundle Status Report Reason Codes

The current Bundle Status Report Reason Codes registry in the Bundle Protocol Namespace is augmented by adding a column identifying the version of the Bundle protocol (Bundle Protocol Version) that applies to the new values. IANA is requested to add the following values, as described in section 6.1.1, to the Bundle Status Report Reason Codes registry. The current values in the Bundle Status Report Reason Codes registry should have the Bundle Protocol Version set to the value 6 or 7 or "6, 7", as shown below.

Bundle Status Report Reason Codes Registry

+-----+-----+-----+				
Bundle	Value	Description	Reference	
Protocol				
Version				
+-----+-----+-----+				
6,7	0	No additional information	[RFC5050],	
			RFC-to-be	
6,7	1	Lifetime expired	[RFC5050],	
			RFC-to-be	
6,7	2	Forwarded over unidirectional	[RFC5050],	
		link	RFC-to-be	
6,7	3	Transmission canceled	[RFC5050],	
			RFC-to-be	

	6,7		4	Depleted storage	[RFC5050],
					RFC-to-be
	6,7		5	Destination endpoint ID	[RFC5050],
				unavailable	RFC-to-be
	6,7		6	No known route to destination	[RFC5050],
				from here	RFC-to-be
	6,7		7	No timely contact with next node	[RFC5050],
				on route	RFC-to-be
	6,7		8	Block unintelligible	[RFC5050],
					RFC-to-be
	7		9	Hop limit exceeded	RFC-to-be
	7		10	Traffic pared	RFC-to-be
	7		11	Block unsupported	RFC-to-be
			12-254	Unassigned	
	6,7		255	Reserved	[RFC6255],
					RFC-to-be
+-----+-----+-----+-----+-----+-----+					

10.6. Bundle Protocol URI scheme types

The Bundle Protocol has a URI scheme type field - an unsigned integer of indefinite length - for which IANA is requested to create and maintain a new "Bundle Protocol URI Scheme Type" registry in the Bundle Protocol Namespace. The "Bundle Protocol URI Scheme Type" registry governs an unsigned integer namespace. Initial values for the Bundle Protocol URI Scheme Type registry are given below.

The registration policy for this registry is: Standards Action. The allocation should only be granted for a standards-track RFC approved by the IESG.

The value range is: unsigned integer.

Each assignment consists of a URI scheme type name and its associated description, a reference to the document that defines the URI scheme, and a reference to the document that defines the use of this URI scheme in BP endpoint IDs (including the CBOR representation of those endpoint IDs in transmitted bundles).

Bundle Protocol URI Scheme Type Registry

+-----+-----+-----+-----+				
		BP Utilization	URI Definition	
Value	Description	Reference	Reference	
+-----+-----+-----+-----+				
0	Reserved	n/a		
1	dtn	RFC-to-be	RFC-to-be	
2	ipn	RFC-to-be	[RFC6260],	
			RFC-to-be	
3-254	Unassigned	n/a		
255-65535	reserved	n/a		
>65535	open for	n/a		
	private use	n/a		
+-----+-----+-----+-----+				

10.7. URI scheme "dtn"

In the Uniform Resource Identifier (URI) Schemes (uri-schemes) registry, IANA is requested to update the registration of the URI scheme with the string "dtn" as the scheme name, as follows:

URI scheme name: "dtn"

Status: permanent

Applications and/or protocols that use this URI scheme name: the Delay-Tolerant Networking (DTN) Bundle Protocol (BP).

Contact:

Scott Burleigh
Jet Propulsion Laboratory,
California Institute of Technology
scott.c.burleigh@jpl.nasa.gov
+1 (800) 393-3353

Change controller:

IETF, iesg@ietf.org

10.8. URI scheme "ipn"

In the Uniform Resource Identifier (URI) Schemes (uri-schemes) registry, IANA is requested to update the registration of the URI scheme with the string "ipn" as the scheme name, originally documented in RFC 6260 [RFC6260], as follows.

URI scheme name: "ipn"

Status: permanent

Applications and/or protocols that use this URI scheme name: the Delay-Tolerant Networking (DTN) Bundle Protocol (BP).

Contact:

Scott Burleigh
Jet Propulsion Laboratory,
California Institute of Technology
scott.c.burleigh@jpl.nasa.gov
+1 (800) 393-3353

Change controller:

IETF, iesg@ietf.org

11. References

11.1. Normative References

[BPSEC] Birrane, E., "Bundle Security Protocol Specification", draft-ietf-dtn-bpsec, January 2020.

[CRC16] ITU-T Recommendation X.25, p. 9, section 2.2.7.4, International Telecommunications Union, October 1996.

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

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.

[RFC8949] Borman, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 8949, December 2020.

[SABR] "Schedule-Aware Bundle Routing", CCSDS Recommended Standard 734.3-B-1, Consultative Committee for Space Data Systems, July 2019.

[TCPCL] Sipos, B., Demmer, M., Ott, J., and S. Perreault, "Delay-Tolerant Networking TCP Convergence Layer Protocol Version 4", draft-ietf-dtn-tcpclv4, January 2020.

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

[URIREG] Thaler, D., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", RFC 7595, BCP 35, June 2015.

11.2. Informative References

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

[BIBE] Burleigh, S., "Bundle-in-Bundle Encapsulation", draft-ietf-dtn-bibect, August 2019.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

[RFC5050] Scott, K. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, November 2007.

[RFC6255] Blanchet, M., "Delay-Tolerant Networking Bundle Protocol IANA Registries", RFC 6255, May 2011.

[RFC6257] Symington, S., Farrell, S., Weiss, H., and P. Lovell, "Bundle Security Protocol Specification", RFC 6257, May 2011.

[RFC6258] Symington, S., "Delay-Tolerant Networking Metadata Extension Block", RFC 6258, May 2011.

[RFC6259] Symington, S., "Delay-Tolerant Networking Previous-Hop Insertion Block", RFC 6259, May 2011.

[RFC6260] Burleigh, S., "Compressed Bundle Header Encoding (CBHE)", RFC 6260, May 2011.

[RFC7143] Chadalapaka, M., Satran, J., Meth, K., and D. Black, "Internet Small Computer System Interface (iSCSI) Protocol (Consolidated)", RFC 7143, April 2014.

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

12. Acknowledgments

This work is freely adapted from RFC 5050, which was an effort of the Delay Tolerant Networking Research Group. The following DTNRG participants contributed significant technical material and/or inputs to that document: 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 Carnegie Mellon University, Stephen Farrell of Trinity College Dublin, Howard Weiss and Peter Lovell of SPARTA, Inc., and Manikantan Ramadas of Ohio University.

This document was prepared using 2-Word-v2.0.template.dot.

13. Significant Changes from RFC 5050

Points on which this draft significantly differs from RFC 5050 include the following:

- . Clarify the difference between transmission and forwarding.
- . Migrate custody transfer to the bundle-in-bundle encapsulation specification [BIBE].
- . Introduce the concept of "node ID" as functionally distinct from endpoint ID, while having the same syntax.
- . Restructure primary block, making it immutable. Add optional CRC.
- . Add optional CRCs to non-primary blocks.
- . Add block ID number to canonical block format (to support BPsec).
- . Add definition of bundle age extension block.
- . Add definition of previous node extension block.
- . Add definition of hop count extension block.
- . Remove Quality of Service markings.
- . Change from SDNVs to CBOR representation.
- . Add lifetime overrides.
- . Time values are denominated in milliseconds, not seconds.

Appendix A.

For More Information

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

Appendix B.

CDDL expression

For informational purposes, Carsten Bormann and Brian Sipos have kindly provided an expression of the Bundle Protocol specification in the Concise Data Definition Language (CDDL). That CDDL expression is presented below. Note that wherever the CDDL expression is in disagreement with the textual representation of the BP specification presented in the earlier sections of this document, the textual representation rules.

```
bpv7_start = bundle / #6.55799(bundle)

; Times before 2000 are invalid
dtn-time = uint

; CRC enumerated type
crc-type = &(
    crc-none: 0,
    crc-16bit: 1,
    crc-32bit: 2
)
; Either 16-bit or 32-bit
crc-value = (bstr .size 2) / (bstr .size 4)

creation-timestamp = [
    dtn-time, ; absolute time of creation
    sequence: uint ; sequence within the time
]
```

```
eid = $eid .within eid-structure
```

```
eid-structure = [
```

```
    uri-code: uint,
```

```
    SSP: any
```

```
]
```

```
$eid /= [
```

```
    uri-code: 1,
```

```
    SSP: (tstr / 0)
```

```
]
```

```
$eid /= [
```

```
    uri-code: 2,
```

```
    SSP: [
```

```
        nodenum: uint,
```

```
        servicenum: uint
```

```
    ]
```

```
]
```

```
; The root bundle array
```

```
bundle = [primary-block, *extension-block, payload-block]
```

```
primary-block = [
```

```
    version: 7,
```

```
    bundle-control-flags,
```

```
    crc-type,
```

```
    destination: eid,
    source-node: eid,
    report-to: eid,
    creation-timestamp,
    lifetime: uint,
    ? (
        fragment-offset: uint,
        total-application-data-length: uint
    ),
    ? crc-value,
]

bundle-control-flags = uint .bits bundleflagbits

bundleflagbits = &(
    reserved: 21,
    reserved: 20,
    reserved: 19,
    bundle-deletion-status-reports-are-requested: 18,
    bundle-delivery-status-reports-are-requested: 17,
    bundle-forwarding-status-reports-are-requested: 16,
    reserved: 15,
    bundle-reception-status-reports-are-requested: 14,
    reserved: 13,
    reserved: 12,
    reserved: 11,
```

```
    reserved: 10,  
    reserved: 9,  
    reserved: 8,  
    reserved: 7,  
    status-time-is-requested-in-all-status-reports: 6,  
    user-application-acknowledgement-is-requested: 5,  
    reserved: 4,  
    reserved: 3,  
    bundle-must-not-be-fragmented: 2,  
    payload-is-an-administrative-record: 1,  
    bundle-is-a-fragment: 0  
)
```

```
; Abstract shared structure of all non-primary blocks  
canonical-block-structure = [  
    block-type-code: uint,  
    block-number: uint,  
    block-control-flags,  
    crc-type,  
    ; Each block type defines the content within the bytestring  
    block-type-specific-data,  
    ? crc-value  
]  
  
block-control-flags = uint .bits blockflagbits
```

```
blockflagbits = &(amp;
    reserved: 7,
    reserved: 6,
    reserved: 5,
    block-must-be-removed-from-bundle-if-it-cannot-be-processed: 4,
    reserved: 3,
    bundle-must-be-deleted-if-block-cannot-be-processed: 2,
    status-report-must-be-transmitted-if-block-cannot-be-processed: 1,
    block-must-be-replicated-in-every-fragment: 0
)

block-type-specific-data = bstr / #6.24(bstr)

; Actual CBOR data embedded in a bytestring, with optional tag to
; indicate so.

; Additional plain bstr allows ciphertext data.

embedded-chor<Item> = (bstr .chor Item) / #6.24(bstr .chor Item) /
bstr

; Extension block type, which does not specialize other than the
; code/number

extension-block = $extension-block .within canonical-block-structure

; Generic shared structure of all non-primary blocks

extension-block-use<CodeValue, BlockData> = [
    block-type-code: CodeValue,
    block-number: (uint .gt 1),
    block-control-flags,
```

```
    crc-type,  
    BlockData,  
    ? crc-value  
]
```

```
; Payload block type
```

```
payload-block = payload-block-structure .within canonical-block-  
structure
```

```
payload-block-structure = [  
    block-type-code: 1,  
    block-number: 1,  
    block-control-flags,  
    crc-type,  
    $payload-block-data,  
    ? crc-value  
]
```

```
; Arbitrary payload data, including non-CBOR bytestring
```

```
$payload-block-data /= block-type-specific-data
```

```
; Administrative record as a payload data specialization
```

```
$payload-block-data /= embedded-cbor<admin-record>
```

```
admin-record = $admin-record .within admin-record-structure
```

```
admin-record-structure = [  
    block-type-code: 1,  
    block-number: 1,  
    block-control-flags,  
    crc-type,  
    $admin-record-data,  
    ? crc-value  
]
```

```
    record-type-code: uint,
    record-content: any
]
; Only one defined record type
$admin-record /= [1, status-record-content]
status-record-content = [
    bundle-status-information,
    status-report-reason-code: uint,
    source-node-eid: eid,
    subject-creation-timestamp: creation-timestamp,
    ? (
        subject-payload-offset: uint,
        subject-payload-length: uint
    )
]
bundle-status-information = [
    reporting-node-received-bundle: status-info-content,
    reporting-node-forwarded-bundle: status-info-content,
    reporting-node-delivered-bundle: status-info-content,
    reporting-node-deleted-bundle: status-info-content
]
status-info-content = [
    status-indicator: bool,
    ? timestamp: dtn-time
```



```
]

; Previous Node extension block

$extension-block /=

    extension-block-use<6, embedded-cbor<ext-data-previous-node>>

ext-data-previous-node = eid


; Bundle Age extension block

$extension-block /=

    extension-block-use<7, embedded-cbor<ext-data-bundle-age>>

ext-data-bundle-age = uint


; Hop Count extension block

$extension-block /=

    extension-block-use<10, embedded-cbor<ext-data-hop-count>>

ext-data-hop-count = [

    hop-limit: uint,

    hop-count: uint

]
```

Authors' Addresses

Scott Burleigh
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109-8099
US
Phone: +1 818 393 3353
Email: Scott.C.Burleigh@jpl.nasa.gov

Kevin Fall
Roland Computing Services
3871 Piedmont Ave. Suite 8
Oakland, CA 94611
US
Email: kfall+rsc@kfall.com

Edward J. Birrane
Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd
Laurel, MD 20723
US
Phone: +1 443 778 7423
Email: Edward.Birrane@jhuapl.edu

Delay-Tolerant Networking
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2017

E. Birrane
K. McKeever
JHU/APL
October 30, 2016

Bundle Protocol Security Specification
draft-ietf-dtn-bpsec-03

Abstract

This document defines a security protocol providing end to end data integrity and confidentiality services for the Bundle Protocol.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 3, 2017.

Copyright Notice

Copyright (c) 2016 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 Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Motivation	3
1.2. Supported Security Services	3
1.3. Specification Scope	4
1.4. Related Documents	5
1.5. Terminology	5
2. Key Properties	7
2.1. Block-Level Granularity	7
2.2. Multiple Security Sources	7
2.3. Mixed Security Policy	8
2.4. User-Selected Ciphersuites	8
2.5. Deterministic Processing	9
3. Security Block Definitions	9
3.1. Block Identification	10
3.2. Block Representation	10
3.3. Block Integrity Block	13
3.4. Block Confidentiality Block	14
3.5. Block Interactions	16
3.6. Parameters and Result Fields	17
3.7. BSP Block Example	18
4. Canonical Forms	20
4.1. Technical Notes	20
4.2. Primary Block Canonicalization	21
4.3. Non-Primary-Block Canonicalization	22
5. Security Processing	22
5.1. Bundles Received from Other Nodes	23
5.1.1. Receiving BCB Blocks	23
5.1.2. Receiving BIB Blocks	23
5.2. Bundle Fragmentation and Reassembly	24
6. Key Management	25
7. Policy Considerations	25
8. Security Considerations	26
8.1. Attacker Capabilities and Objectives	27
8.2. Attacker Behaviors and BPsec Mitigations	28
8.2.1. Eavesdropping Attacks	28
8.2.2. Modification Attacks	28
8.2.3. Topology Attacks	29
8.2.4. Message Injection	30
9. Ciphersuite Authorship Considerations	30
10. Defining Other Security Blocks	31
11. Conformance	32
12. IANA Considerations	32
12.1. Bundle Block Types	32
12.2. Cipher Suite Flags	32
12.3. Parameters and Results	33
13. References	34

13.1. Normative References	34
13.2. Informative References	34
Appendix A. Acknowledgements	35
Authors' Addresses	35

1. Introduction

This document defines security features for the Bundle Protocol [BPBIS] intended for use in delay-tolerant networks, in order to provide Delay-Tolerant Networking (DTN) security services.

1.1. Motivation

The Bundle Protocol is used in DTNs that overlay multiple networks, some of which may be challenged by limitations such as intermittent and possibly unpredictable loss of connectivity, long or variable delay, asymmetric data rates, and high error rates. The purpose of the Bundle Protocol is to support interoperability across such stressed networks.

The stressed environment of the underlying networks over which the Bundle Protocol operates makes it important for the DTN to be protected from unauthorized use, and this stressed environment poses unique challenges for the mechanisms needed to secure the Bundle Protocol. Furthermore, DTNs may be deployed in environments where a portion of the network might become compromised, posing the usual security challenges related to confidentiality and integrity.

1.2. Supported Security Services

This specification supports end-to-end integrity and confidentiality services associated with BP bundles.

Integrity services ensure data within a bundle are not changed. Data changes may be caused by processing errors, environmental conditions, or intentional manipulation. An integrity service is one that provides sufficient confidence to a data receiver that data has not changed since its value was last asserted.

Confidentiality services ensure that the values of some data within a bundle can only be determined by authorized receivers of the data. When a bundle traverses a DTN, many nodes in the network other than the destination node MAY see the contents of a bundle. A confidentiality service allows a destination node to generate data values from otherwise encrypted contents of a bundle.

NOTE: Hop-by-hop authentication is NOT a supported security service in this specification, for three reasons.

1. The term "hop-by-hop" is ambiguous in a BP overlay, as nodes that are adjacent in the overlay may not be adjacent in physical connectivity. This condition is difficult or impossible to predict in the overlay and therefore makes the concept of hop-by-hop authentication difficult or impossible to enforce at the overlay.
2. Networks in which BPsec may be deployed may have a mixture of security-aware and not-security-aware nodes. Hop-by-hop authentication cannot be deployed in a network if adjacent nodes in the network have different security capabilities.
3. Hop-by-hop authentication can be viewed as a special case of data integrity. As such, it is possible to develop policy that provides a version of authentication using the integrity mechanisms defined in this specification.

1.3. Specification Scope

This document describes the Bundle Protocol Security Specification (BPsec), which provides security services for blocks within a bundle. This includes the data specification for individual BP extension blocks and the processing instructions for those blocks.

BPsec applies, by definition, only to those nodes that implement it, known as "security-aware" nodes. There MAY be other nodes in the DTN that do not implement BPsec. All nodes can interoperate with the exception that BPsec security operations can only happen at BPsec security-aware nodes.

This specification does not address individual cipher suite implementations. The definition and enumeration of cipher suites should be undertaken in separate specification documents.

This specification does not address the implementation of security policy and does not provide a security policy for the BPsec. Security policies are typically based on the nature and capabilities of individual networks and network operational concepts. However, this specification does recommend policy considerations when building a security policy.

This specification does not address how to combine the BPsec security blocks with other protocols, other BP extension blocks, or other best practices to achieve security in any particular network implementation.

1.4. Related Documents

This document is best read and understood within the context of the following other DTN documents:

"Delay-Tolerant Networking Architecture" [RFC4838] defines the architecture for delay-tolerant networks, but does not discuss security at any length.

The DTN Bundle Protocol [BPBIS] defines the format and processing of the blocks used to implement the Bundle Protocol, excluding the security-specific blocks defined here.

The Bundle Security Protocol [RFC6257] and Streamlined Bundle Security Protocol [SBSP] introduce the concepts of security blocks for security services. BPsec is based off of these documents.

1.5. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This section defines those terms whose definition is important to the understanding of concepts within this specification.

- o Source - the bundle node from which a bundle originates.
- o Destination - the bundle node to which a bundle is ultimately destined.
- o Forwarder - the bundle node that forwarded the bundle on its most recent hop.
- o Intermediate Receiver, Waypoint, or "Next Hop" - the neighboring bundle node to which a forwarder forwards a bundle.
- o Path - the ordered sequence of nodes through which a bundle passes on its way from source to destination. The path is not necessarily known by the bundle, or any bundle-aware nodes.

The application of these terms applied to a sample network topology is shown in Figure 1. This figure shows four bundle nodes (BN1, BN2, BN3, BN4) residing above some transport layer(s). Three distinct transport and network protocols (T1/N1, T2/N2, and T3/N3) are also shown.

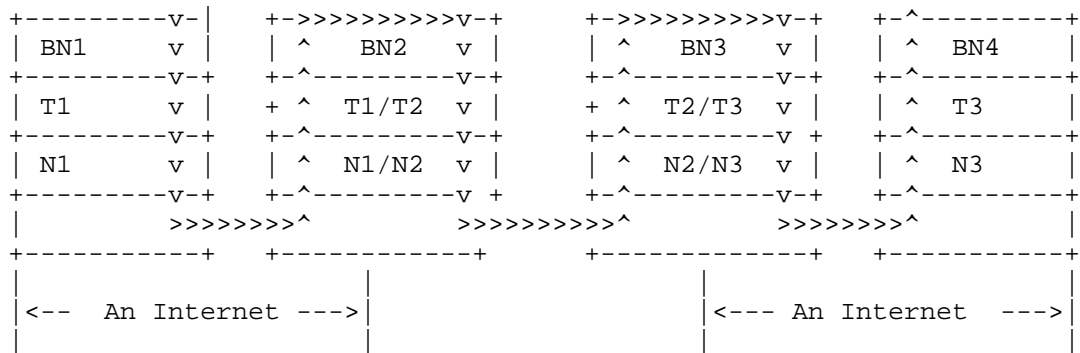


Figure 1: Bundle Nodes Sitting Above the Transport Layer.

Consider the case where BN1 originates a bundle that it forwards to BN2. BN2 forwards the bundle to BN3, and BN3 forwards the bundle to BN4. BN1 is the source of the bundle and BN4 is the destination of the bundle. BN1 is the first forwarder, and BN2 is the first intermediate receiver; BN2 then becomes the forwarder, and BN3 the intermediate receiver; BN3 then becomes the last forwarder, and BN4 the last intermediate receiver, as well as the destination.

If node BN2 originates a bundle (for example, a bundle status report or a custodial signal), which is then forwarded on to BN3, and then to BN4, then BN2 is the source of the bundle (as well as being the first forwarder of the bundle) and BN4 is the destination of the bundle (as well as being the final intermediate receiver).

The following security-specific terminology is also defined to clarify security operations in this specification.

- o Security Service - the security features supported by this specification: integrity and confidentiality.
- o Security Source - a bundle node that adds a security block to a bundle.
- o Security Target - the block within a bundle that receives a security-service as part of a security-operation.
- o Security Block - a BPSec extension block in a bundle.
- o Security Operation - the application of a security service to a security target, notated as OP(security service, security target). For example, OP(confidentiality, payload). Every security

operation in a bundle MUST be unique, meaning that a security service can only be applied to a security target once in a bundle. A security operation is implemented by a security block.

2. Key Properties

The application of security services in a DTN is a complex endeavor that must consider physical properties of the network, policies at each node, and various application security requirements. Rather than enumerate all potential security implementations in all potential DTN topologies, this specification defines a set of key properties of a security system. The security primitives outlined in this document MUST enable the realization of these properties in a DTN deploying the Bundle Protocol.

2.1. Block-Level Granularity

Blocks within a bundle represent different types of information. The primary block contains identification and routing information. The payload block carries application data. Extension blocks carry a variety of data that may augment or annotate the payload, or otherwise provide information necessary for the proper processing of a bundle along a path. Therefore, applying a single level and type of security across an entire bundle fails to recognize that blocks in a bundle may represent different types of information with different security needs.

Security services within this specification MUST provide block level granularity where applicable such that different blocks within a bundle may have different security services applied to them.

For example, within a bundle, a payload might be encrypted to protect its contents, whereas an extension block containing summary information related to the payload might be integrity signed but otherwise unencrypted to provide certain nodes access to payload-related data without providing access to the payload.

Each security block in a bundle will be associated with a specific security operation.

2.2. Multiple Security Sources

A bundle MAY have multiple security blocks and these blocks MAY have different security sources.

The Bundle Protocol allows extension blocks to be added to a bundle at any time during its existence in the DTN. When a waypoint node adds a new extension block to a bundle, that extension block may have

security services applied to it by that waypoint. Similarly, a waypoint node may add a security service to an existing extension block, consistent with its security policy. For example, a node representing a boundary between a trusted part of the network and an untrusted part of the network may wish to apply payload encryption for bundles leaving the trusted portion of the network.

In each case, a node other than the bundle originator may add a security service to the bundle and, as such, the source for the security service will be different than the source of the bundle itself. Security services **MUST** track their originating node so as to properly apply policy and key selection associated with processing the security service at the bundle destination.

Referring to Figure 1, if the bundle that originates at BN1 is given security blocks by BN1, then BN1 is the security source for those blocks as well as being the source of the bundle. If the bundle that originates at BN1 is then given a security block by BN2, then BN2 is the security source for that block even though BN1 remains the bundle source.

2.3. Mixed Security Policy

Different nodes in a DTN may have different security related capabilities. Some nodes may not be security aware and will not understand any security related extension blocks. Other nodes may have security policies that require evaluation of security services at places other than the bundle destination (such as verifying integrity signatures at certain waypoint nodes). Other nodes may ignore any security processing if they are not the destination of the bundle. The security services described in this specification must allow each of these scenarios.

Extension blocks representing security services **MUST** have their block processing flags set such that the block will be treated appropriately by non-security-aware nodes.

Extension blocks providing integrity services within a bundle **MUST** support options to allow waypoint nodes to evaluate these signatures if such nodes have the proper configuraton to do so.

2.4. User-Selected Ciphersuites

The security services defined in this specification rely on a variety of cipher suites providing integrity signatures, ciphertext, and other information necessary to populate security blocks. Users may wish to select different cipher suites to implement different security services. For example, some users may wish to use a SHA-256

based hash for integrity whereas other users may require a SHA-384 hash instead. The security services defined in this specification MUST provide a mechanism for identifying what cipher suite has been used to populate a security block.

2.5. Deterministic Processing

In all cases, the processing order of security services within a bundle must avoid ambiguity when evaluating security at the bundle destination. This specification MUST provide determinism in the application and evaluation of security services, even when doing so results in a loss of flexibility.

3. Security Block Definitions

There are two types of security blocks that may be included in a bundle. These are the Block Integrity Block (BIB) and the Block Confidentiality Block (BCB).

The BIB is used to ensure the integrity of its security target(s). The integrity information in the BIB MAY (when possible) be verified by any node in between the BIB security source and the bundle destination. BIBs MAY be added to, and removed from, bundles as a matter of security policy.

The BCB indicates that the security target(s) has been encrypted, in whole or in part, at the BCB security source in order to protect its content while in transit. The BCB may be decrypted by appropriate nodes in the network, up to and including the bundle destination, as a matter of security policy.

A security operation MUST NOT be applied more than once in a bundle. For example, the two security operations: OP(integrity, payload) and OP(integrity, payload) are considered redundant and MUST NOT appear together in a bundle. However, the two security operations OP(integrity, payload) and OP(integrity, extension_block_1) MAY both be present in the bundle. Also, the two security operations OP(integrity, extension_block_1) and OP(integrity, extension_block_2) are unique and may both appear in the same bundle.

If the same security service is to be applied to multiple security targets, and cipher suite parameters for each security service are identical, then the set of security operations can be represented as a single security block with multiple security targets. In such a case, all security operations represented in the security block MUST be applied/evaluated together.

3.1. Block Identification

This specification requires that every target block of a security operation be uniquely identifiable. The definition of the extension block header from [BPBIS] provides such a mechanism in the "Block Number" field, which provides a unique identifier for a block within a bundle. Within this specification, a security target will be identified by its unique Block Number.

A security block MAY apply to multiple security targets if and only if all cipher suite parameters, security source, and key information are common for the security operation. In such a case, the security block MUST contain security results for each covered security target. The use of multiple security targets in a security block provides an efficiency mechanism so that identical ciphersuite information does not need to be repeated across multiple security blocks.

3.2. Block Representation

Each security block uses the Canonical Bundle Block Format as defined in [BPBIS]. That is, each security block is comprised of the following elements:

- o Block Type Code
- o Block Number
- o Block Processing Control Flags
- o CRC Type and CRC Field
- o Block Data Length
- o Block Type Specific Data Fields

The structure of the BIB and BCB Block Type Specific Data fields are identical and illustrated in Figure 2. In this figure, field names prefaced with an '*' are optional and their inclusion in the block is indicated by the Cipher Suite Flags field.

Field Name	Field Data Type
# Security Targets	Unsigned Integer
Security Targets	Array (Unsigned Integer)
Cipher Suite ID	Unsigned Integer
Cipher Suite Flags	Unsigned Integer
Security Source	URI - OPTIONAL
Cipher Parameters	Byte Array - OPTIONAL
Security Result	Byte Array

Figure 2: BIB and BCB Block Structure

Where the block fields are identified as follows.

- o # Security Targets - The number of security targets for this security block. This value MUST be at least 1.
- o Security Targets - This array contains the unique identifier of the blocks targetted by this security operation. Each security target MUST represent a block present in the bundle. A security target MUST NOT be repeated in this array.
- o Cipher suite ID - Identifies the cipher suite used to implement the security service represented by this block and applied to each security target.
- o Cipher suite flags - Identifies which optional security block fields are present in the block. The structure of the Cipher Suite Flags field is shown in Figure 3. The presence of an optional field is indicated by setting the value of the corresponding flag to one. A value of zero indicates the corresponding optional field is not present. The BPSEC Cipher Suite Flags are defined as follows.

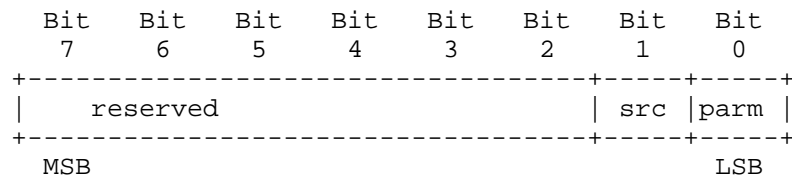


Figure 3: Cipher Suite Flags

Where:

- * bits 7-2 are reserved for future use.
 - * src - bit 1 indicates whether the Security Source is present in the block.
 - * parm - bit 0 indicates whether or not the Cipher Suite Parameters field is present in the block.
 - o (OPTIONAL) Security Source (URI) - This identifies the node that inserted the security service in the bundle. If the security source is not present then the source MAY be inferred from the bundle source, the previous hop, or some other node as defined by security policy.
 - o (OPTIONAL) Parameters (Byte Array) - Compound field of the following two items.
 - * Length (Unsigned Integer) - specifies the length of the next field, which captures the parameters data.
 - * Data (Byte Array) - A byte array encoding one or more cipher suite parameters, with each parameter represented as a Type-Length-Value (TLV) triplet, defined as follows.
 - + Type (Byte) - The parameter type.
 - + Length (Unsigned Integer) - The length of the parameter.
 - + Value (Byte Array) - The parameter value.
- See Section 3.6 for a list of parameter types that MUST be supported by BPSEC implementations. BPSEC cipher suite specifications MAY define their own parameters to be represented in this byte array.
- o Security Result (Byte Array) - A security result is the output of an appropriate cipher suite specific calculation (e.g., a

signature, Message Authentication Code (MAC), or cipher-text block key). There MUST exist one security result for each security target in the security block. A security result is a multi-field component, described as follows.

- * Total Length (Unsigned Integer) - specifies the length, in bytes, of the remaining security result information.
- * Results (Byte Array) - This field captures each of the security results, catenated together, one for each security target covered by the security block. Each result is captured by the four-tuple of (Target, Type, Len, Value). The meaning of each is given below.
 - + Target (Optional) (Unsigned Integer) - If the security block has multiple security targets, the target field is the Block Number of the security target to which this result field applies. If the security block only has a single security target, this field is omitted.
 - + Type (Unsigned Integer) - The type of security result field.
 - + Length (Unsigned Integer) - The length of the result field.
 - + Value (Byte Array) - The results of the cipher suite specific calculation.

3.3. Block Integrity Block

A BIB is an ASB with the following characteristics:

The Block Type Code value MUST be 0x02.

The Block Processing Control flags value can be set to whatever values are required by local policy. Cipher suite designers should carefully consider the effect of setting flags that either discard the block or delete the bundle in the event that this block cannot be processed.

A security target for a BIB MUST NOT reference a security block defined in this specification (e.g., a BIB or a BCB).

The cipher suite ID MUST be documented as an end-to-end authentication-cipher suite or as an end-to-end error-detection-cipher suite.

An EID-reference to the security source MAY be present. If this field is not present, then the security source of the block SHOULD

be inferred according to security policy and MAY default to the bundle source. The security source may also be specified as part of key information described in Section 3.6.

The security result captures the result of applying the cipher suite calculation (e.g., the MAC or signature) to the relevant parts of the security target, as specified in the cipher suite definition. This field **MUST** be present.

The cipher suite MAY process less than the entire security target. If the cipher suite processes less than the complete, original security target, the cipher suite parameters **MUST** specify which bytes of the security target are protected.

Notes:

- o Since OP(integrity, target) is allowed only once in a bundle per target, it is **RECOMMENDED** that users wishing to support multiple integrity signatures for the same target define a multi-signature cipher suite.
- o For some cipher suites, (e.g., those using asymmetric keying to produce signatures or those using symmetric keying with a group key), the security information MAY be checked at any hop on the way to the destination that has access to the required keying information, in accordance with Section 3.5.
- o The use of a generally available key is **RECOMMENDED** if custodial transfer is employed and all nodes **SHOULD** verify the bundle before accepting custody.

3.4. Block Confidentiality Block

A BCB is an ASB with the following characteristics:

The Block Type Code value **MUST** be 0x03.

The Block Processing Control flags value can be set to whatever values are required by local policy, except that this block **MUST** have the "replicate in every fragment" flag set if the target of the BCB is the Payload Block. Having that BCB in each fragment indicates to a receiving node that the payload portion of each fragment represents cipher-text. Cipher suite designers should carefully consider the effect of setting flags that either discard the block or delete the bundle in the event that this block cannot be processed.

A security target for a BCB MAY reference the payload block, a non-security extension block, or a BIB block. A security target in a BCB MUST NOT be another BCB.

The cipher suite ID MUST be documented as a confidentiality cipher suite.

Any additional bytes generated as a result of encryption and/or authentication processing of the security target SHOULD be placed in an "integrity check value" field (see Section 3.6) or other such appropriate area in the security result of the BCB.

An EID-reference to the security source MAY be present. If this field is not present, then the security source of the block SHOULD be inferred according to security policy and MAY default to the bundle source. The security source may also be specified as part of key information described in Section 3.6.

The security result MUST be present in the BCB. This compound field normally contains fields such as an encrypted bundle encryption key and/or authentication tag.

The BCB modifies the contents of its security target. When a BCB is applied, the security target body data are encrypted "in-place". Following encryption, the security target body data contains ciphertext, not plain-text. Other security target block fields (such as type, processing control flags, and length) remain unmodified.

Fragmentation, reassembly, and custody transfer are adversely affected by a change in size of the payload due to ambiguity about what byte range of the block is actually in any particular fragment. Therefore, when the security target of a BCB is the bundle payload, the BCB MUST NOT alter the size of the payload block body data. Cipher suites SHOULD place any block expansion, such as authentication tags (integrity check values) and any padding generated by a block-mode cipher, into an integrity check value item in the security result field (see Section 3.6) of the BCB. This "in-place" encryption allows fragmentation, reassembly, and custody transfer to operate without knowledge of whether or not encryption has occurred.

Notes:

- o The cipher suite MAY process less than the entire original security target body data. If the cipher suite processes less than the complete, original security target body data, the BCB for that security target MUST specify, as part of the cipher suite parameters, which bytes of the body data are protected.

- o The BCB's "discard" flag may be set independently from its security target's "discard" flag. Whether or not the BCB's "discard" flag is set is an implementation/policy decision for the encrypting node. (The "discard" flag is more properly called the "Discard if block cannot be processed" flag.)
- o A BCB MAY include information as part of additional authenticated data to address parts of the target block, such as EID references, that are not converted to cipher-text.

3.5. Block Interactions

The security block types defined in this specification are designed to be as independent as possible. However, there are some cases where security blocks may share a security target creating processing dependencies.

If confidentiality is being applied to a target that already has integrity applied to it, then an undesirable condition occurs where a security aware intermediate node would be unable to check the integrity result of a block because the block contents have been encrypted after the integrity signature was generated. To address this concern, the following processing rules MUST be followed.

- o If confidentiality is to be applied to a target, it MUST also be applied to any integrity operation already defined for that target. This means that if a BCB is added to encrypt a block, another BCB MUST also be added to encrypt a BIB also targeting that block.
- o An integrity operation MUST NOT be applied to a security target if a BCB in the bundle shares the same security target. This prevents ambiguity in the order of evaluation when receiving a BIB and a BCB for a given security target.
- o An integrity value MUST NOT be evaluated if the BIB providing the integrity value is the security target of an existing BCB block in the bundle. In such a case, the BIB data contains cipher-text as it has been encrypted.
- o An integrity value MUST NOT be evaluated if the security target of the BIB is also the security target of a BCB in the bundle. In such a case, the security target data contains cipher-text as it has been encrypted.
- o As mentioned in Section 3.3, a BIB MUST NOT have a BCB as its security target. BCBs may embed integrity results as part of cipher suite parameters.

These restrictions on block interactions impose a necessary ordering when applying security operations within a bundle. Specifically, for a given security target, BIBs MUST be added before BCBs. This ordering MUST be preserved in cases where the current BPA is adding all of the security blocks for the bundle or whether the BPA is a waypoint adding new security blocks to a bundle that already contains security blocks.

3.6. Parameters and Result Fields

Various cipher suites include several items in the cipher suite parameters and/or security result fields. Which items MAY appear is defined by the particular cipher suite description. A cipher suite MAY support several instances of the same type within a single block.

Each item is represented as a type-length-value. Type is a single byte indicating the item. Length is the count of data bytes to follow, and is an Unsigned Integer. Value is the data content of the item.

Item types, name, and descriptions are defined as follows.

Cipher suite parameters and result fields.

Type	Name	Description	Field
0	Reserved		
1	Initialization Vector (IV)	A random value, typically eight to sixteen bytes.	Cipher Suite Parameters
2	Reserved		
3	Key Information	Material encoded or protected by the key management system and used to transport an ephemeral key protected by a long-term key.	Cipher Suite Parameters
4	Content Range	Pair of Unsigned Integers (offset,length) specifying the range of payload bytes to which an operation applies. The offset MUST be the offset within the	Cipher Suite Parameters

		original bundle, even if the current bundle is a fragment.	
5	Integrity Signatures	Result of BIB digest or other signing operation.	Security Results
6	Unassigned		
7	Salt	An IV-like value used by certain confidentiality suites.	Cipher Suite Parameters
8	BCB Integrity Check Value (ICV) / Authentication Tag	Output from certain confidentiality cipher suite operations to be used at the destination to verify that the protected data has not been modified. This value MAY contain padding if required by the cipher suite.	Security Results
9-255	Reserved		

Table 1

3.7. BSP Block Example

An example of BPsec blocks applied to a bundle is illustrated in Figure 4. In this figure the first column represents blocks within a bundle and the second column represents a unique identifier for each block, suitable for use as the security target of a BPsec security block. Since the mechanism and format of a security target is not specified in this document, the terminology B1...Bn is used to identify blocks in the bundle for the purposes of illustration.

Block in Bundle	ID
Primary Block	B1
BIB OP(integrity, target=B1)	B2
BCB OP(confidentiality, target=B4)	B3
Extension Block	B4
BIB OP(integrity, target=B6)	B5
Extension Block	B6
BCB OP(confidentiality, target=B8, B9)	B7
BIB (encrypted by B7) OP(integrity, target=B9)	B8
Payload Block	B9

Figure 4: Sample Use of BSP Blocks

In this example a bundle has four non-security-related blocks: the primary block (B1), three extension blocks (B4, B6), and a payload block (B9). The following security applications are applied to this bundle.

- o An integrity signature applied to the canonicalized primary block. This is accomplished by a single BIB (B2).
- o Confidentiality for the first extension block (B4). This is accomplished by a BCB block (B3).
- o Integrity for the second extension block (B6). This is accomplished by a BIB block (B5). NOTE: If the extension block B6 contains a representation of the serialized bundle (such as a hash over all blocks in the bundle at the time of its last transmission) then the BIB block is also providing an authentication service from the prior BPSEC-BPA to this BPSEC-BPA.
- o An integrity signature on the payload (B10). This is accomplished by a BIB block (B8).

- o Confidentiality for the payload block and it's integrity signature. This is accomplished by a BCB block, B7, encrypting B8 and B9.

4. Canonical Forms

By definition, an integrity service determines whether any aspect of a block was changed from the moment the security service was applied at the security source until the point of current evaluation. To successfully verify the integrity of a block, the data passed to the verifying cipher suite MUST be the same bits, in the same order, as those passed to the signature-generating cipher suite at the security source.

However, [BPBIS] does not specify a single on-the-wire encoding of bundles. In cases where a security source generates a different encoding than that used at a receiving node, care MUST be taken to ensure that the inputs to cipher suites at the receiving node is a bitwise match to inputs provided at the security source.

This section provides guidance on how to create a canonical form for each type of block in a bundle. This form MUST be used when generating inputs to cipher suites for use by BPSec blocks.

This specification does not define any security operation over the entire bundle and, therefore, provides no canonical form for a serialized bundle.

4.1. Technical Notes

The following technical considerations hold for all canonicalizations in this section.

- o Any numeric fields defined as variable-length MUST be expanded to their "unpacked" form. For example, a 32-bit integer value MUST be unpacked to a four-byte representation.
- o Each block encoding MUST follow the CBOR encodings provided in [BPBISCBOR].
- o Canonical forms are not transmitted, they are used to generate input to a cipher suite for security processing at a security-aware node.
- o Reserved flags MUST NOT be included in any canonicalization as it is not known if those flags will change in transit.

- o These canonicalization algorithms assume that endpoint IDs themselves are immutable and they are unsuitable for use in environments where that assumption might be violated.
- o Cipher suites MAY define their own canonicalization algorithms and require the use of those algorithms over the ones provided in this specification. In the event of conflicting canonicalization algorithms, cipher suite algorithms take precedence over this specification.

4.2. Primary Block Canonicalization

The primary block canonical form is the same as the CBOR encoding of the block, with certain modifications to account for allowed block changes as the bundle traverses the DTN. The fields that compromise the primary block, and any special considerations for their representation in a canonical form, are as follows.

- o The Version field is included, without modification.
- o The Bundle Processing Flags field is used, with modification. Certain bundle processing flags MAY change as a bundle transits the DTN without indicating an integrity error. These flags, which are identified below, MUST NOT be represented in the canonicalized form of the bundle processing flags and, instead, be represented by the bit 0.
 - * Reserved flags.
 - * Bundle is a Fragment flag.
- o The CRC Type, Destination EID, Source Node ID, Report-To EID, Creation Timestamp, and Lifetime fields are included, without modification.
- o The fragment ID field MAY change if the bundle is fragmented in transit and, as such, this field MUST NOT be included in the canonicalization.
- o The CRC field MAY change at each hop - for example, if a bundle becomes fragmented, each fragment will have a different CRC value from the original signed primary block. As such, this field MUST NOT be included in the canonicalization.

4.3. Non-Primary-Block Canonicalization

All non-primary blocks (NPBs) in [BPBIS] share the same block structure and should be canonicalized in the same way.

Canonicalization for NPBs is dependent on whether the security operation being performed is integrity or confidentiality. Integrity operations consider every field in the block, whereas confidentiality operations only consider the block-type-specific data. Since confidentiality is applied to hide information (replacing plaintext with ciphertext) it provides no benefit to include in the confidentiality calculation information that **MUST** remain readable, such as block fields other than the block-type-specific data.

The fields that comprise a NPB, and any special considerations for their representation in a canonical form, are as follows.

- o The Block Type Code field is included, without modification, for integrity operations and omitted for confidentiality operations.
- o The Block Number field is included, without modification, for integrity operations and omitted for confidentiality operations.
- o The Block Processing Control Flags field is included, without modification, for integrity operations and omitted for confidentiality operations, with the exception of reserved flags which are treated as 0 in both cases.
- o The CRC type and CRC fields are included, without modification, for integrity operations and omitted for confidentiality operations.
- o The Block Type Specific Data field is included, without modification, for both integrity and confidentiality operations, with the exception that in some cases only a portion of the payload data is to be processed. In such a case, only those bytes are included in the canonical form and additional cipher suite parameters are required to specify which part of the field is included.

5. Security Processing

This section describes the security aspects of bundle processing.

5.1. Bundles Received from Other Nodes

Security blocks MUST be processed in a specific order when received by a security-aware node. The processing order is as follows.

- o All BCB blocks in the bundle MUST be evaluated prior to evaluating any BIBs in the bundle. When BIBs and BCBs share a security target, BCBs MUST be evaluated first and BIBs second.

5.1.1. Receiving BCB Blocks

If a received bundle contains a BCB, the receiving node MUST determine whether it has the responsibility of decrypting the BCB security target and removing the BCB prior to delivering data to an application at the node or forwarding the bundle.

If the receiving node is the destination of the bundle, the node MUST decrypt any BCBs remaining in the bundle. If the receiving node is not the destination of the bundle, the node MAY decrypt the BCB if directed to do so as a matter of security policy.

If the relevant parts of an encrypted payload block cannot be decrypted (i.e., the decryption key cannot be deduced or decryption fails), then the bundle MUST be discarded and processed no further. If an encrypted security target other than the payload block cannot be decrypted then the associated security target and all security blocks associated with that target MUST be discarded and processed no further. In both cases, requested status reports (see [BPBIS]) MAY be generated to reflect bundle or block deletion.

When a BCB is decrypted, the recovered plain-text MUST replace the cipher-text in the security target body data

If a BCB contains multiple security targets, all security targets MUST be processed if the BCB is processed by the Node. The effect of this is to be the same as if each security target had been represented by an individual BCB with a single security target.

5.1.2. Receiving BIB Blocks

If a received bundle contains a BIB, the receiving node MUST determine whether it has the responsibility of verifying the BIB security target and whether to remove the BIB prior to delivering data to an application at the node or forwarding the bundle.

A BIB MUST NOT be processed if the security target of the BIB is also the security target of a BCB in the bundle. Given the order of operations mandated by this specification, when both a BIB and a BCB

share a security target, it means that the security target MUST have been encrypted after it was integrity signed and, therefore, the BIB cannot be verified until the security target has been decrypted by processing the BCB.

If the security policy of a security-aware node specifies that a bundle should have applied integrity to a specific security target and no such BIB is present in the bundle, then the node MUST process this security target in accordance with the security policy. This MAY involve removing the security target from the bundle. If the removed security target is the payload or primary block, the bundle MAY be discarded. This action may occur at any node that has the ability to verify an integrity signature, not just the bundle destination.

If the bundle has a BIB and the receiving node is the destination for the bundle, the node MUST verify the security target in accordance with the cipher suite specification. If a BIB check fails, the security target has failed to authenticate and the security target SHALL be processed according to the security policy. A bundle status report indicating the failure MAY be generated. Otherwise, if the BIB verifies, the security target is ready to be processed for delivery.

If the bundle has a BIB and the receiving node is not the bundle destination, the receiving node MAY attempt to verify the value in the security result field. If the check fails, the node SHALL process the security target in accordance to local security policy. It is RECOMMENDED that if a payload integrity check fails at a waypoint that it is processed in the same way as if the check fails at the destination.

If a BIB contains multiple security targets, all security targets MUST be processed if the BIB is processed by the Node. The effect of this is to be the same as if each security target had been represented by an individual BIB with a single security target.

5.2. Bundle Fragmentation and Reassembly

If it is necessary for a node to fragment a bundle and security services have been applied to that bundle, the fragmentation rules described in [BPBIS] MUST be followed. As defined there and repeated here for completeness, only the payload may be fragmented; security blocks, like all extension blocks, can never be fragmented.

Due to the complexity of bundle fragmentation, including the possibility of fragmenting bundle fragments, integrity and confidentiality operations are not to be applied to a bundle

representing a fragment (i.e., a bundle whose "bundle is a Fragment" flag is set in the Bundle Processing Control Flags field). Specifically, a BCB or BIB MUST NOT be added to a bundle fragment, even if the security target of the security block is not the payload. When integrity and confidentiality must be applied to a fragment, we RECOMMEND that encapsulation be used instead.

6. Key Management

Key management in delay-tolerant networks is recognized as a difficult topic and is one that this specification does not attempt to solve.

7. Policy Considerations

When implementing BPsec, several policy decisions must be considered. This section describes key policies that affect the generation, forwarding, and receipt of bundles that are secured using this specification.

- o If a bundle is received that contains more than one security operation, in violation of BPsec, then the BPA must determine how to handle this bundle. The bundle may be discarded, the block affected by the security operation may be discarded, or one security operation may be favored over another.
- o BPAs in the network MUST understand what security operations they should apply to bundles. This decision may be based on the source of the bundle, the destination of the bundle, or some other information related to the bundle.
- o If an intermediate receiver has been configured to add a security operation to a bundle, and the received bundle already has the security operation applied, then the receiver MUST understand what to do. The receiver may discard the bundle, discard the security target and associated BPsec blocks, replace the security operation, or some other action.
- o It is recommended that security operations only be applied to the payload block, the primary block, and any block-types specifically identified in the security policy. If a BPA were to apply security operations such as integrity or confidentiality to every block in the bundle, regardless of the block type, there could be downstream errors processing blocks whose contents must be inspected at every hop in the network path.

- o Adding a BIB to a security target that has already been encrypted by a BCB is not allowed. Therefore, we recommend three methods to add an integrity signature to an encrypted security target.
 - 1. At the time of encryption, an integrity signature may be generated and added to the BCB for the security target as additional information in the security result field.
 - 2. The encrypted block may be replicated as a new block and integrity signed.
 - 3. An encapsulation scheme may be applied to encapsulate the security target (or the entire bundle) such that the encapsulating structure is, itself, no longer the security target of a BCB and may therefore be the security target of a BIB.

8. Security Considerations

Given the nature of delay-tolerant networking applications, it is expected that bundles may traverse a variety of environments and devices which each pose unique security risks and requirements on the implementation of security within BPSEC. For these reasons, it is important to introduce key threat models and describe the roles and responsibilities of the BPSEC protocol in protecting the confidentiality and integrity of the data against those threats throughout the DTN. This section provides additional discussion on security threats that BPSEC will face and describe in additional detail how BPSEC security mechanisms operate to mitigate these threats.

It should be noted that BPSEC addresses only the security of data traveling over the DTN, not the underlying DTN itself. Additionally, BPSEC addresses neither the fitness of externally-defined cryptographic methods nor the security of their implementation. It is the responsibility of the BPSEC implementer that appropriate algorithms and methods are chosen. Furthermore, the BPSEC protocol does not address threats which share computing resources with the DTN and/or BPSEC software implementations. These threats may be malicious software or compromised libraries which intend to intercept data or recover cryptographic material. Here, it is the responsibility of the BPSEC implementer to ensure that any cryptographic material, including shared secret or private keys, is protected against access within both memory and storage devices.

The threat model described here is assumed to have a set of capabilities identical to those described by the Internet Threat Model in [RFC3552], but the BPSEC threat model is scoped to

illustrate threats specific to BPSEC operating within DTN environments and therefore focuses on man-in-the-middle (MITM) attackers.

8.1. Attacker Capabilities and Objectives

BPSEC was designed to protect against MITM threats which may have access to a bundle during transit from its source, Alice, to its destination, Bob. A MITM node, Mallory, is a non-cooperative node operating on the DTN between Alice and Bob that has the ability to receive bundles, examine bundles, modify bundles, forward bundles, and generate bundles at will in order to compromise the confidentiality or integrity of data within the DTN. For the purposes of this section, any MITM node is assumed to effectively be security-aware even if it does not implement the BPSEC protocol. There are three classes of MITM nodes which are differentiated based on their access to cryptographic material:

- o Unprivileged Node: Mallory has not been provisioned within the secure environment and only has access to cryptographic material which has been publicly-shared.
- o Legitimate Node: Mallory is within the secure environment and therefore has access to cryptographic material which has been provisioned to Mallory (i.e., K_M) as well as material which has been publicly-shared.
- o Privileged Node: Mallory is a privileged node within the secure environment and therefore has access to cryptographic material which has been provisioned to Mallory, Alice and/or Bob (i.e., K_M , K_A , and/or K_B) as well as material which has been publicly-shared.

If Mallory is operating as a privileged node, this is tantamount to compromise; BPSEC does not provide mechanisms to detect or remove Mallory from the DTN or BPSEC secure environment. It is up to the BPSEC implementer or the underlying cryptographic mechanisms to provide appropriate capabilities if they are needed. It should also be noted that if the implementation of BPSEC uses a single set of shared cryptographic material for all nodes, a legitimate node is equivalent to a privileged node because $K_M == K_A == K_B$.

A special case of the legitimate node is when Mallory is either Alice or Bob (i.e., $K_M == K_A$ or $K_M == K_B$). In this case, Mallory is able to impersonate traffic as either Alice or Bob, which means that traffic to and from that node can be decrypted and encrypted, respectively. Additionally, messages may be signed as originating from one of the endpoints.

8.2. Attacker Behaviors and BPsec Mitigations

8.2.1. Eavesdropping Attacks

Once Mallory has received a bundle, she is able to examine the contents of that bundle and attempt to recover any protected data or cryptographic keying material from the blocks contained within. The protection mechanism that BPsec provides against this action is the BCB, which encrypts the contents of its security target, providing confidentiality of the data. Of course, it should be assumed that Mallory is able to attempt offline recovery of encrypted data, so the cryptographic mechanisms selected to protect the data should provide a suitable level of protection.

When evaluating the risk of eavesdropping attacks, it is important to consider the lifetime of bundles on a DTN. Depending on the network, bundles may persist for days or even years. If a bundle does persist on the network for years and the cipher suite used for a BCB provides inadequate protection, Mallory may be able to recover the protected data before that bundle reaches its intended destination.

8.2.2. Modification Attacks

As a node participating in the DTN between Alice and Bob, Mallory will also be able to modify the received bundle, including non-BPsec data such as the primary block, payload blocks, or block processing control flags as defined in [BPBIS]. Mallory will be able to undertake activities which include modification of data within the blocks, replacement of blocks, addition of blocks, or removal of blocks. Within BPsec, both the BIB and BCB provide integrity protection mechanisms to detect or prevent data manipulation attempts by Mallory.

The BIB provides that protection to another block which is its security target. The cryptographic mechanisms used to generate the BIB should be strong against collision attacks and Mallory should not have access to the cryptographic material used by the originating node to generate the BIB (e.g., K_A). If both of these conditions are true, Mallory will be unable to modify the security target or the BIB and lead Bob to validate the security target as originating from Alice.

Since BPsec security operations are implemented by placing blocks in a bundle, there is no in-band mechanism for detecting or correcting certain cases where Mallory removes blocks from a bundle. If Mallory removes a BCB block, but keeps the security target, the security target remains encrypted and there is a possibility that there may no longer be sufficient information to decrypt the block at its

destination. If Mallory removes both a BCB (or BIB) and its security target there is no evidence left in the bundle of the security operation. Similarly, if Mallory removes the BIB but not the security target there is no evidence left in the bundle of the security operation. In each of these cases, the implementation of BPSec MUST be combined with policy configuration at endpoints in the network which describe the expected and required security operations that must be applied on transmission and are expected to be present on receipt. This or other similar out-of-band information is required to correct for removal of security information in the bundle.

A limitation of the BIB may exist within the implementation of BIB validation at the destination node. If Mallory is a legitimate node within the DTN, the BIB generated by Alice with K_A can be replaced with a new BIB generated with K_M and forwarded to Bob. If Bob is only validating that the BIB was generated by a legitimate user, Bob will acknowledge the message as originating from Mallory instead of Alice. In order to provide verifiable integrity checks, both a BIB and BCB should be used. Alice creates a BIB with the protected data block as the security target and then creates a BCB with both the BIB and protected data block as its security targets. In this configuration, since Mallory is only a legitimate node and does not have access to Alice's key K_A , Mallory is unable to decrypt the BCB and replace the BIB.

8.2.3. Topology Attacks

If Mallory is in a MITM position within the DTN, she is able to influence how any bundles that come to her may pass through the network. Upon receiving and processing a bundle that must be routed elsewhere in the network, Mallory has three options as to how to proceed: not forward the bundle, forward the bundle as intended, or forward the bundle to one or more specific nodes within the network.

Attacks that involve re-routing the packets throughout the network are essentially a special case of the modification attacks described in this section where the attacker is modifying fields within the primary block of the bundle. Given that BPSec cannot encrypt the contents of the primary block, alternate methods must be used to prevent this situation. These methods MAY include requiring BIBs for primary blocks, using encapsulation, or otherwise strategically manipulating primary block data. The specifics of any such mitigation technique are specific to the implementation of the deploying network and outside of the scope of this document.

Furthermore, routing rules and policies may be useful in enforcing particular traffic flows to prevent topology attacks. While these

rules and policies may utilize some features provided by BPSec, their definition is beyond the scope of this specification.

8.2.4. Message Injection

Mallory is also able to generate new bundles and transmit them into the DTN at will. These bundles may either be copies or slight modifications of previously-observed bundles (i.e., a replay attack) or entirely new bundles generated based on the Bundle Protocol, BPSec, or other bundle-related protocols. With these attacks Mallory's objectives may vary, but may be targeting either the bundle protocol or application-layer protocols conveyed by the bundle protocol.

BPSec relies on cipher suite capabilities to prevent replay or forged message attacks. A BCB used with appropriate cryptographic mechanisms (e.g., a counter-based cipher mode) may provide replay protection under certain circumstances. Alternatively, application data itself may be augmented to include mechanisms to assert data uniqueness and then protected with a BIB, a BCB, or both along with other block data. In such a case, the receiving node would be able to validate the uniqueness of the data.

9. Ciphersuite Authorship Considerations

Cipher suite developers or implementers should consider the diverse performance and conditions of networks on which the Bundle Protocol (and therefore BPSec) will operate. Specifically, the delay and capacity of delay-tolerant networks can vary substantially. Cipher suite developers should consider these conditions to better describe the conditions when those suites will operate or exhibit vulnerability, and selection of these suites for implementation should be made with consideration to the reality. There are key differences that may limit the opportunity to leverage existing cipher suites and technologies that have been developed for use in traditional, more reliable networks:

- o Data Lifetime: Depending on the application environment, bundles may persist on the network for extended periods of time, perhaps even years. Cryptographic algorithms should be selected to ensure protection of data against attacks for a length of time reasonable for the application.
- o One-Way Traffic: Depending on the application environment, it is possible that only a one-way connection may exist between two endpoints, or if a two-way connection does exist, the round-trip time may be extremely large. This may limit the utility of

session key generation mechanisms, such as Diffie-Hellman, as a two-way handshake may not be feasible or reliable.

- o Opportunistic Access: Depending on the application environment, a given endpoint may not be guaranteed to be accessible within a certain amount of time. This may make asymmetric cryptographic architectures which rely on a key distribution center or other trust center impractical under certain conditions.

10. Defining Other Security Blocks

Other security blocks (OSBs) may be defined and used in addition to the security blocks identified in this specification. Both the usage of BIB, BCB, and any future OSBs MAY co-exist within a bundle and MAY be considered in conformance with BPsec if each of the following requirements are met by any future identified security blocks.

- o Other security blocks (OSBs) MUST NOT reuse any enumerations identified in this specification, to include the block type codes for BIB and BCB.
- o An OSB definition MUST state whether it can be the target of a BIB or a BCB. The definition MUST also state whether the OSB can target a BIB or a BCB.
- o An OSB definition MUST provide a deterministic processing order in the event that a bundle is received containing BIBs, BCBs, and OSBs. This processing order MUST NOT alter the BIB and BCB processing orders identified in this specification.
- o An OSB definition MUST provide a canonicalization algorithm if the default non-primary-block canonicalization algorithm cannot be used to generate a deterministic input for a cipher suite. This requirement MAY be waived if the OSB is defined so as to never be the security target of a BIB or a BCB.
- o An OSB definition MAY NOT require any behavior of a BPSEC-BPA that is in conflict with the behavior identified in this specification. In particular, the security processing requirements imposed by this specification MUST be consistent across all BPSEC-BPAs in a network.
- o The behavior of an OSB when dealing with fragmentation MUST be specified and MUST NOT lead to ambiguous processing states. In particular, an OSB definition should address how to receive and process an OSB in a bundle fragment that may or may not also contain its security target. An OSB definition should also

address whether an OSB may be added to a bundle marked as a fragment.

Additionally, policy considerations for the management, monitoring, and configuration associated with blocks SHOULD be included in any OSB definition.

NOTE: The burden of showing compliance with processing rules is placed upon the standards defining new security blocks and the identification of such blocks shall not, alone, require maintenance of this specification.

11. Conformance

All implementations are strongly RECOMMENDED to provide some method of hop-by-hop verification by generating a hash to some canonical form of the bundle and placing an integrity signature on that form using a BIB.

12. IANA Considerations

This protocol has fields that have been registered by IANA.

12.1. Bundle Block Types

This specification allocates three block types from the existing "Bundle Block Types" registry defined in [RFC6255] .

Additional Entries for the Bundle Block-Type Codes Registry:

Value	Description	Reference
2	Block Integrity Block	This document
3	Block Confidentiality Block	This document

Table 2

12.2. Cipher Suite Flags

This protocol has a cipher suite flags field and certain flags are defined. An IANA registry has been set up as follows.

The registration policy for this registry is: Specification Required

The Value range is: Variable Length

Cipher Suite Flag Registry:

Bit Position (right to left)	Description	Reference
0	Block contains result	This document
1	Block Contains parameters	This document
2	Source EID ref present	This document
>3	Reserved	This document

Table 3

12.3. Parameters and Results

This protocol has fields for cipher suite parameters and results. The field is a type-length-value triple and a registry is required for the "type" sub-field. The values for "type" apply to both the cipher suite parameters and the cipher suite results fields. Certain values are defined. An IANA registry has been set up as follows.

The registration policy for this registry is: Specification Required

The Value range is: 8-bit unsigned integer.

Cipher Suite Parameters and Results Type Registry:

Value	Description	Reference
0	reserved	Section 3.6
1	initialization vector (IV)	Section 3.6
2	reserved	Section 3.6
3	key information	Section 3.6
4	content-range (pair of Unsigned Integers)	Section 3.6
5	integrity signature	Section 3.6
6	unassigned	Section 3.6
7	salt	Section 3.6
8	BCB integrity check value (ICV)	Section 3.6
9-191	reserved	Section 3.6
192-250	private use	Section 3.6
251-255	reserved	Section 3.6

Table 4

13. References

13.1. Normative References

- [BPBIS] Burleigh, S., Fall, K., and E. Birrane, "Bundle Protocol", draft-ietf-dtn-bpbis-04 (work in progress), July 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<http://www.rfc-editor.org/info/rfc3552>>.
- [RFC6255] Blanchet, M., "Delay-Tolerant Networking Bundle Protocol IANA Registries", RFC 6255, May 2011.

13.2. Informative References

- [BPBISCBOR] Burleigh, S., "Bundle Protocol CBOR Representation Specification", draft-burleigh-dtn-rs-cbor-01 (work in progress), April 2016.

- [RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, April 2007.
- [RFC6257] Symington, S., Farrell, S., Weiss, H., and P. Lovell, "Bundle Security Protocol Specification", RFC 6257, May 2011.
- [SBSP] Birrane, E., "Streamlined Bundle Security Protocol", draft-birrane-dtn-sbsp-01 (work in progress), October 2015.

Appendix A. Acknowledgements

The following participants contributed technical material, use cases, and useful thoughts on the overall approach to this security specification: Scott Burleigh of the Jet Propulsion Laboratory, Amy Alford and Angela Hennessy of the Laboratory for Telecommunications Sciences, and Angela Dalton and Cherita Corbett of the Johns Hopkins University Applied Physics Laboratory.

Authors' Addresses

Edward J. Birrane, III
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
US

Phone: +1 443 778 7423
Email: Edward.Birrane@jhuapl.edu

Kenneth McKeever
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
US

Phone: +1 443 778 2237
Email: Ken.McKeever@jhuapl.edu

Delay-Tolerant Networking
Internet-Draft
Intended status: Standards Track
Expires: August 19, 2021

E. Birrane
K. McKeever
JHU/APL
February 15, 2021

Bundle Protocol Security Specification
draft-ietf-dtn-bpsec-27

Abstract

This document defines a security protocol providing data integrity and confidentiality services for the Bundle Protocol.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 19, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Supported Security Services	3
1.2.	Specification Scope	4
1.3.	Related Documents	5
1.4.	Terminology	6
2.	Design Decisions	7
2.1.	Block-Level Granularity	7
2.2.	Multiple Security Sources	8
2.3.	Mixed Security Policy	9
2.4.	User-Defined Security Contexts	9
2.5.	Deterministic Processing	9
3.	Security Blocks	10
3.1.	Block Definitions	10
3.2.	Uniqueness	10
3.3.	Target Multiplicity	12
3.4.	Target Identification	13
3.5.	Block Representation	13
3.6.	Abstract Security Block	13
3.7.	Block Integrity Block	16
3.8.	Block Confidentiality Block	17
3.9.	Block Interactions	18
3.10.	Parameter and Result Identification	19
3.11.	BSP Block Examples	20
3.11.1.	Example 1: Constructing a Bundle with Security	20
3.11.2.	Example 2: Adding More Security At A New Node	21
4.	Canonical Forms	23
5.	Security Processing	24
5.1.	Bundles Received from Other Nodes	24
5.1.1.	Receiving BCBs	24
5.1.2.	Receiving BIBs	25
5.2.	Bundle Fragmentation and Reassembly	26
6.	Key Management	27
7.	Security Policy Considerations	27
7.1.	Security Reason Codes	28
8.	Security Considerations	30
8.1.	Attacker Capabilities and Objectives	30
8.2.	Attacker Behaviors and BPSec Mitigations	31
8.2.1.	Eavesdropping Attacks	31
8.2.2.	Modification Attacks	32
8.2.3.	Topology Attacks	33
8.2.4.	Message Injection	34
9.	Security Context Considerations	34
9.1.	Mandating Security Contexts	34
9.2.	Identification and Configuration	35
9.3.	Authorship	37
10.	Defining Other Security Blocks	38

11. IANA Considerations	39
11.1. Bundle Block Types	39
11.2. Bundle Status Report Reason Codes	40
11.3. Security Context Identifiers	40
12. References	41
12.1. Normative References	41
12.2. Informative References	42
Appendix A. Acknowledgements	42
Authors' Addresses	42

1. Introduction

This document defines security features for the Bundle Protocol (BP) [I-D.ietf-dtn-bpbis] and is intended for use in Delay Tolerant Networks (DTNs) to provide security services between a security source and a security acceptor. When the security source is the bundle source and when the security acceptor is the bundle destination, the security service provides end-to-end protection.

The Bundle Protocol specification [I-D.ietf-dtn-bpbis] defines DTN as referring to "a networking architecture providing communications in and/or through highly stressed environments" where "BP may be viewed as sitting at the application layer of some number of constituent networks, forming a store-carry-forward overlay network". The term "stressed" environment refers to multiple challenging conditions including intermittent connectivity, large and/or variable delays, asymmetric data rates, and high bit error rates.

It should be presumed that the BP will be deployed such that the network cannot be trusted, posing the usual security challenges related to confidentiality and integrity. However, the stressed nature of the BP operating environment imposes unique conditions where usual transport security mechanisms may not be sufficient. For example, the store-carry-forward nature of the network may require protecting data at rest, preventing unauthorized consumption of critical resources such as storage space, and operating without regular contact with a centralized security oracle (such as a certificate authority).

An end-to-end security service is needed that operates in all of the environments where the BP operates.

1.1. Supported Security Services

BPSec provides integrity and confidentiality services for BP bundles, as defined in this section.

Integrity services ensure that changes to target data within a bundle can be discovered. Data changes may be caused by processing errors, environmental conditions, or intentional manipulation. In the context of BPSec, integrity services apply to plain text in the bundle.

Confidentiality services ensure that target data is unintelligible to nodes in the DTN, except for authorized nodes possessing special information. This generally means producing cipher text from plain text and generating authentication information for that cipher text. Confidentiality, in this context, applies to the contents of target data and does not extend to hiding the fact that confidentiality exists in the bundle.

NOTE: Hop-by-hop authentication is NOT a supported security service in this specification, for two reasons.

1. The term "hop-by-hop" is ambiguous in a BP overlay, as nodes that are adjacent in the overlay may not be adjacent in physical connectivity. This condition is difficult or impossible to detect and therefore hop-by-hop authentication is difficult or impossible to enforce.
2. Hop-by-hop authentication cannot be deployed in a network if adjacent nodes in the network have incompatible security capabilities.

1.2. Specification Scope

This document defines the security services provided by the BPSec. This includes the data specification for representing these services as BP extension blocks, and the rules for adding, removing, and processing these blocks at various points during the bundle's traversal of the DTN.

BPSec addresses only the security of data traveling over the DTN, not the underlying DTN itself. Furthermore, while the BPSec protocol can provide security-at-rest in a store-carry-forward network, it does not address threats which share computing resources with the DTN and/or BPSec software implementations. These threats may be malicious software or compromised libraries which intend to intercept data or recover cryptographic material. Here, it is the responsibility of the BPSec implementer to ensure that any cryptographic material, including shared secret or private keys, is protected against access within both memory and storage devices.

Completely trusted networks are extremely uncommon. Amongst untrusted networks, different networking conditions and operational

considerations require varying strengths of security mechanism. Mandating a single security context may result in too much security for some networks and too little security in others. It is expected that separate documents define different security contexts for use in different networks. A set of default security contexts are defined in ([I-D.ietf-dtn-bpsec-default-sc]) and provide basic security services for interoperability testing and for operational use on the terrestrial Internet.

This specification addresses neither the fitness of externally-defined cryptographic methods nor the security of their implementation.

This specification does not address the implementation of security policy and does not provide a security policy for the BPsec. Similar to cipher suites, security policies are based on the nature and capabilities of individual networks and network operational concepts. This specification does provide policy considerations when building a security policy.

With the exception of the Bundle Protocol, this specification does not address how to combine the BPsec security blocks with other protocols, other BP extension blocks, or other best practices to achieve security in any particular network implementation.

1.3. Related Documents

This document is best read and understood within the context of the following other DTN documents:

"Delay-Tolerant Networking Architecture" [RFC4838] defines the architecture for DTNs and identifies certain security assumptions made by existing Internet protocols that are not valid in a DTN.

The Bundle Protocol [I-D.ietf-dtn-bpbis] defines the format and processing of bundles, defines the extension block format used to represent BPsec security blocks, and defines the canonical block structure used by this specification.

The Concise Binary Object Representation (CBOR) format [RFC8949] defines a data format that allows for small code size, fairly small message size, and extensibility without version negotiation. The block-specific-data associated with BPsec security blocks are encoded in this data format.

The Bundle Security Protocol [RFC6257] and Streamlined Bundle Security Protocol [I-D.birrane-dtn-sbsp] documents introduced the

concepts of using BP extension blocks for security services in a DTN. The BPsec is a continuation and refinement of these documents.

1.4. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This section defines terminology either unique to the BPsec or otherwise necessary for understanding the concepts defined in this specification.

- o Bundle Destination - the node which receives a bundle and delivers the payload of the bundle to an application. Also, the Node ID of the Bundle Protocol Agent (BPA) receiving the bundle. The bundle destination acts as the security acceptor for every security target in every security block in every bundle it receives.
- o Bundle Source - the node which originates a bundle. Also, the Node ID of the BPA originating the bundle.
- o Cipher Suite - a set of one or more algorithms providing integrity and/or confidentiality services. Cipher suites may define user parameters (e.g. secret keys to use) but do not provide values for those parameters.
- o Forwarder - any node that transmits a bundle in the DTN. Also, the Node ID of the BPA that sent the bundle on its most recent hop.
- o Intermediate Receiver, Waypoint, or Next Hop - any node that receives a bundle from a Forwarder that is not the Bundle Destination. Also, the Node ID of the BPA at any such node.
- o Path - the ordered sequence of nodes through which a bundle passes on its way from Source to Destination. The path is not necessarily known in advance by the bundle or any BPAs in the DTN.
- o Security Acceptor - a bundle node that processes and dispositions one or more security blocks in a bundle. Security acceptors act as the endpoint of a security service represented in a security block. They remove the security blocks they act upon as part of processing and disposition. Also, the Node ID of that node.
- o Security Block - a BPsec extension block in a bundle.

- o Security Context - the set of assumptions, algorithms, configurations and policies used to implement security services.
- o Security Operation - the application of a given security service to a security target, notated as OP(security service, security target). For example, OP(bcb-confidentiality, payload). Every security operation in a bundle MUST be unique, meaning that a given security service can only be applied to a security target once in a bundle. A security operation is implemented by a security block.
- o Security Service - a process that gives some protection to a security target. For example, this specification defines security services for plain text integrity (bib-integrity), and authenticated plain text confidentiality with additional authenticated data (bcb-confidentiality).
- o Security Source - a bundle node that adds a security block to a bundle. Also, the Node ID of that node.
- o Security Target - the block within a bundle that receives a security service as part of a security operation.
- o Security Verifier - a bundle node that verifies the correctness of one or more security blocks in a bundle. Unlike security acceptors, security verifiers do not act as the endpoint of a security service and do not remove verified security blocks. Also, the Node ID of that node.

2. Design Decisions

The application of security services in a DTN is a complex endeavor that must consider physical properties of the network (such as connectivity and propagation times), policies at each node, application security requirements, and current and future threat environments. This section identifies those desirable properties that guide design decisions for this specification and are necessary for understanding the format and behavior of the BPSec protocol.

2.1. Block-Level Granularity

Security services within this specification must allow different blocks within a bundle to have different security services applied to them.

Blocks within a bundle represent different types of information. The primary block contains identification and routing information. The payload block carries application data. Extension blocks carry a

variety of data that may augment or annotate the payload, or otherwise provide information necessary for the proper processing of a bundle along a path. Therefore, applying a single level and type of security across an entire bundle fails to recognize that blocks in a bundle represent different types of information with different security needs.

For example, a payload block might be encrypted to protect its contents and an extension block containing summary information related to the payload might be integrity signed but unencrypted to provide waypoints access to payload-related data without providing access to the payload.

2.2. Multiple Security Sources

A bundle can have multiple security blocks and these blocks can have different security sources. BPsec implementations **MUST NOT** assume that all blocks in a bundle have the same security operations applied to them.

The Bundle Protocol allows extension blocks to be added to a bundle at any time during its existence in the DTN. When a waypoint adds a new extension block to a bundle, that extension block **MAY** have security services applied to it by that waypoint. Similarly, a waypoint **MAY** add a security service to an existing block, consistent with its security policy.

When a waypoint adds a security service to the bundle, the waypoint is the security source for that service. The security block(s) which represent that service in the bundle may need to record this security source as the bundle destination might need this information for processing.

For example, a bundle source may choose to apply an integrity service to its plain text payload. Later a waypoint node, representing a gateway to another portion of the DTN, may receive the bundle and choose to apply a confidentiality service. In this case, the integrity security source is the bundle source and the confidentiality security source is the waypoint node.

In cases where the security source and security acceptor are not the bundle source and bundle destination, it is possible that the bundle will reach the bundle destination prior to reaching a security acceptor. In cases where this may be a practical problem, it is recommended that solutions such as bundle encapsulation can be used to ensure that a bundle be delivered to a security acceptor prior to being delivered to the bundle destination. Generally, if a bundle reaches a waypoint that has the appropriate configuration and policy

to act as a security acceptor for a security service in the bundle, then the waypoint should act as that security acceptor.

2.3. Mixed Security Policy

The security policy enforced by nodes in the DTN may differ.

Some waypoints will have security policies that require evaluating security services even if they are not the bundle destination or the final intended acceptor of the service. For example, a waypoint could choose to verify an integrity service even though the waypoint is not the bundle destination and the integrity service will be needed by other nodes along the bundle's path.

Some waypoints will determine, through policy, that they are the intended recipient of the security service and terminate the security service in the bundle. For example, a gateway node could determine that, even though it is not the destination of the bundle, it should verify and remove a particular integrity service or attempt to decrypt a confidentiality service, before forwarding the bundle along its path.

Some waypoints could understand security blocks but refuse to process them unless they are the bundle destination.

2.4. User-Defined Security Contexts

A security context is the union of security algorithms (cipher suites), policies associated with the use of those algorithms, and configuration values. Different contexts may specify different algorithms, different policies, or different configuration values used in the implementation of their security services. BPsec provides a mechanism to define security contexts. Users may select from registered security contexts and customize those contexts through security context parameters.

For example, some users might prefer a SHA2 hash function for integrity whereas other users might prefer a SHA3 hash function. Providing either separate security contexts or a single, parameterized security context allows users flexibility in applying the desired cipher suite, policy, and configuration when populating a security block.

2.5. Deterministic Processing

Whenever a node determines that it must process more than one security block in a received bundle (either because the policy at a waypoint states that it should process security blocks or because the

node is the bundle destination) the order in which security blocks are processed must be deterministic. All nodes must impose this same deterministic processing order for all security blocks. This specification provides determinism in the application and evaluation of security services, even when doing so results in a loss of flexibility.

3. Security Blocks

3.1. Block Definitions

This specification defines two types of security block: the Block Integrity Block (BIB) and the Block Confidentiality Block (BCB).

The BIB is used to ensure the integrity of its plain text security target(s). The integrity information in the BIB MAY be verified by any node along the bundle path from the BIB security source to the bundle destination. Waypoints add or remove BIBs from bundles in accordance with their security policy. BIBs are never used for integrity protection of the cipher text provided by a BCB. Because security policy at BPsec nodes may differ regarding integrity verification, BIBs do not guarantee hop-by-hop authentication, as discussed in Section 1.1.

The BCB indicates that the security target(s) have been encrypted at the BCB security source in order to protect their content while in transit. The BCB is decrypted by security acceptor nodes in the network, up to and including the bundle destination, as a matter of security policy. BCBs additionally provide integrity protection mechanisms for the cipher text they generate.

3.2. Uniqueness

Security operations in a bundle MUST be unique; the same security service MUST NOT be applied to a security target more than once in a bundle. Since a security operation is represented by a security block, this means that multiple security blocks of the same type cannot share the same security targets. A new security block MUST NOT be added to a bundle if a pre-existing security block of the same type is already defined for the security target of the new security block.

This uniqueness requirement ensures that there is no ambiguity related to the order in which security blocks are processed or how security policy can be specified to require certain security services be present in a bundle.

Using the notation `OP(service, target)`, several examples illustrate this uniqueness requirement.

- o Signing the payload twice: The two operations `OP(bib-integrity, payload)` and `OP(bib-integrity, payload)` are redundant and MUST NOT both be present in the same bundle at the same time.
- o Signing different blocks: The two operations `OP(bib-integrity, payload)` and `OP(bib-integrity, extension_block_1)` are not redundant and both may be present in the same bundle at the same time. Similarly, the two operations `OP(bib-integrity, extension_block_1)` and `OP(bib-integrity, extension_block_2)` are also not redundant and may both be present in the bundle at the same time.
- o Different Services on same block: The two operations `OP(bib-integrity, payload)` and `OP(bcb-confidentiality, payload)` are not inherently redundant and may both be present in the bundle at the same time, pursuant to other processing rules in this specification.
- o Different services from different block types: The notation `OP(service, target)` refers specifically to a security block, as the security block is the embodiment of a security service applied to a security target in a bundle. Were some Other Security Block (OSB) to be defined providing an integrity service, then the operations `OP(bib-integrity, target)` and `OP(osb-integrity, target)` MAY both be present in the same bundle if so allowed by the definition of the OSB, as discussed in Section 10.

NOTES:

A security block may be removed from a bundle as part of security processing at a waypoint node with a new security block being added to the bundle by that node. In this case, conflicting security blocks never co-exist in the bundle at the same time and the uniqueness requirement is not violated.

A cipher text integrity mechanism (such as associated authenticated data) calculated by a cipher suite and transported in a BCB is considered part of the confidentiality service and, therefore, unique from the plain text integrity service provided by a BIB.

The security blocks defined in this specification (BIB and BCB) are designed with the intention that the BPA adding these blocks is the authoritative source of the security service. If a BPA adds a BIB on a security target, then the BIB is expected to be

the authoritative source of integrity for that security target. If a BPA adds a BCB to a security target, then the BCB is expected to be the authoritative source of confidentiality for that security target. More complex scenarios, such as having multiple nodes in a network sign the same security target, can be accommodated using the definition of custom security contexts (Section 9) and/or the definition of other security blocks (Section 10).

3.3. Target Multiplicity

A single security block MAY represent multiple security operations as a way of reducing the overall number of security blocks present in a bundle. In these circumstances, reducing the number of security blocks in the bundle reduces the amount of redundant information in the bundle.

A set of security operations can be represented by a single security block when all of the following conditions are true.

- o The security operations apply the same security service. For example, they are all integrity operations or all confidentiality operations.
- o The security context parameters for the security operations are identical.
- o The security source for the security operations is the same, meaning the set of operations are being added by the same node.
- o No security operations have the same security target, as that would violate the need for security operations to be unique.
- o None of the security operations conflict with security operations already present in the bundle.

When representing multiple security operations in a single security block, the information that is common across all operations is represented once in the security block, and the information which is different (e.g., the security targets) are represented individually.

It is RECOMMENDED that if a node processes any security operation in a security block that it process all security operations in the security block. This allows security sources to assert that the set of security operations in a security block are expected to be processed by the same security acceptor. However, the determination of whether a node actually is a security acceptor or not is a matter of the policy of the node itself. In cases where a receiving node

determines that it is the security acceptor of only a subset of the security operations in a security block, the node may choose to only process that subset of security operations.

3.4. Target Identification

A security target is a block in the bundle to which a security service applies. This target must be uniquely and unambiguously identifiable when processing a security block. The definition of the extension block header from [I-D.ietf-dtn-bpbis] provides a "Block Number" field suitable for this purpose. Therefore, a security target in a security block MUST be represented as the Block Number of the target block.

3.5. Block Representation

Each security block uses the Canonical Bundle Block Format as defined in [I-D.ietf-dtn-bpbis]. That is, each security block is comprised of the following elements:

- o block type code
- o block number
- o block processing control flags
- o CRC type
- o block-type-specific-data
- o CRC field (if present)

Security-specific information for a security block is captured in the block-type-specific-data field.

3.6. Abstract Security Block

The structure of the security-specific portions of a security block is identical for both the BIB and BCB Block Types. Therefore, this section defines an Abstract Security Block (ASB) data structure and discusses the definition, processing, and other constraints for using this structure. An ASB is never directly instantiated within a bundle, it is only a mechanism for discussing the common aspects of BIB and BCB security blocks.

The fields of the ASB SHALL be as follows, listed in the order in which they must appear. The encoding of these fields MUST be in accordance with the canonical forms provided in Section 4.

Security Targets:

This field identifies the block(s) targeted by the security operation(s) represented by this security block. Each target block is represented by its unique Block Number. This field SHALL be represented by a CBOR array of data items. Each target within this CBOR array SHALL be represented by a CBOR unsigned integer. This array MUST have at least 1 entry and each entry MUST represent the Block Number of a block that exists in the bundle. There MUST NOT be duplicate entries in this array. The order of elements in this list has no semantic meaning outside of the context of this block. Within the block, the ordering of targets must match the ordering of results associated with these targets.

Security Context Id:

This field identifies the security context used to implement the security service represented by this block and applied to each security target. This field SHALL be represented by a CBOR unsigned integer. The values for this Id should come from the registry defined in Section 11.3

Security Context Flags:

This field identifies which optional fields are present in the security block. This field SHALL be represented as a CBOR unsigned integer whose contents shall be interpreted as a bit field. Each bit in this bit field indicates the presence (bit set to 1) or absence (bit set to 0) of optional data in the security block. The association of bits to security block data is defined as follows.

Bit 0 (the least-significant bit, 0x01): Security Context Parameters Present Flag.

Bit >0 Reserved

Implementations MUST set reserved bits to 0 when writing this field and MUST ignore the values of reserved bits when reading this field. For unreserved bits, a value of 1 indicates that the associated security block field MUST be included in the security block. A value of 0 indicates that the associated security block field MUST NOT be in the security block.

Security Source:

This field identifies the Endpoint that inserted the security block in the bundle. This field SHALL be represented by a CBOR array in accordance with [I-D.ietf-dtn-bpbis] rules for representing Endpoint Identifiers (EIDs).

Security Context Parameters (Optional):

This field captures one or more security context parameters that should be used when processing the security service described by this security block. This field SHALL be represented by a CBOR array. Each entry in this array is a single security context parameter. A single parameter SHALL also be represented as a CBOR array comprising a 2-tuple of the id and value of the parameter, as follows.

- * Parameter Id. This field identifies which parameter is being specified. This field SHALL be represented as a CBOR unsigned integer. Parameter Ids are selected as described in Section 3.10.
- * Parameter Value. This field captures the value associated with this parameter. This field SHALL be represented by the applicable CBOR representation of the parameter, in accordance with Section 3.10.

The logical layout of the parameters array is illustrated in Figure 1.

Parameter 1		Parameter 2		...	Parameter N	
Id	Value	Id	Value		Id	Value

Figure 1: Security Context Parameters

Security Results:

This field captures the results of applying a security service to the security targets of the security block. This field SHALL be represented as a CBOR array of target results. Each entry in this array represents the set of security results for a specific security target. The target results MUST be ordered identically to the Security Targets field of the security block. This means that the first set of target results in this array corresponds to the first entry in the Security Targets field of the security block, and so on. There MUST be one entry in this array for each entry in the Security Targets field of the security block.

The set of security results for a target is also represented as a CBOR array of individual results. An individual result is represented as a 2-tuple of a result id and a result value, defined as follows.

- * Result Id. This field identifies which security result is being specified. Some security results capture the primary output of a cipher suite. Other security results contain additional annotative information from cipher suite processing. This field SHALL be represented as a CBOR unsigned integer. Security result Ids will be as specified in Section 3.10.
- * Result Value. This field captures the value associated with the result. This field SHALL be represented by the applicable CBOR representation of the result value, in accordance with Section 3.10.

The logical layout of the security results array is illustrated in Figure 2. In this figure there are N security targets for this security block. The first security target contains M results and the Nth security target contains K results.

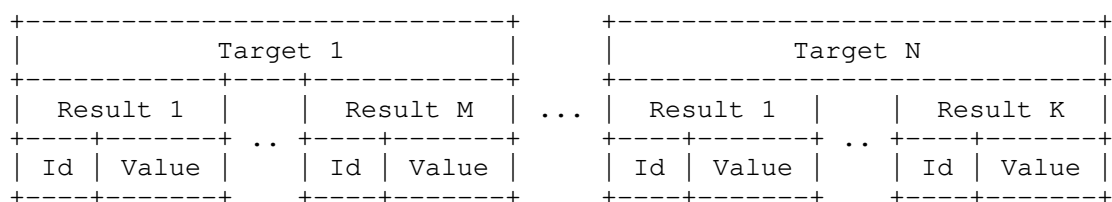


Figure 2: Security Results

3.7. Block Integrity Block

A BIB is a bundle extension block with the following characteristics.

The Block Type Code value is as specified in Section 11.1.

The block-type-specific-data field follows the structure of the ASB.

A security target listed in the Security Targets field MUST NOT reference a security block defined in this specification (e.g., a BIB or a BCB).

The Security Context MUST utilize an authentication mechanism or an error detection mechanism.

Notes:

- o Designers SHOULD carefully consider the effect of setting flags that either discard the block or delete the bundle in the event that this block cannot be processed.
- o Since OP(bib-integrity, target) is allowed only once in a bundle per target, it is RECOMMENDED that users wishing to support multiple integrity mechanisms for the same target define a multi-result security context. Such a context could generate multiple security results for the same security target using different integrity-protection mechanisms or different configurations for the same integrity-protection mechanism.
- o A BIB is used to verify the plain text integrity of its security target. However, a single BIB MAY include security results for blocks other than its security target when doing so establishes a needed relationship between the BIB security target and other blocks in the bundle (such as the primary block).
- o Security information MAY be checked at any hop on the way to the bundle destination that has access to the required keying information, in accordance with Section 3.9.

3.8. Block Confidentiality Block

A BCB is a bundle extension block with the following characteristics.

The Block Type Code value is as specified in Section 11.1.

The Block Processing Control flags value can be set to whatever values are required by local policy with the following exceptions. BCB blocks MUST have the "block must be replicated in every fragment" flag set if one of the targets is the payload block. Having that BCB in each fragment indicates to a receiving node that the payload portion of each fragment represents cipher text. BCB blocks MUST NOT have the "block must be removed from bundle if it can't be processed" flag set. Removing a BCB from a bundle without decrypting its security targets removes information from the bundle necessary for their later decryption.

The block-type-specific-data fields follow the structure of the ASB.

A security target listed in the Security Targets field can reference the payload block, a non-security extension block, or a BIB. A BCB MUST NOT include another BCB as a security target. A BCB MUST NOT target the primary block. A BCB MUST NOT target a BIB block unless it shares a security target with that BIB block.

Any Security Context used by a BCB MUST utilize a confidentiality cipher that provides authenticated encryption with associated data (AEAD).

Additional information created by a cipher suite (such as an authentication tag) can be placed either in a security result field or in the generated cipher text. The determination of where to place this information is a function of the cipher suite and security context used.

The BCB modifies the contents of its security target(s). When a BCB is applied, the security target body data are encrypted "in-place". Following encryption, the security target block-type-specific-data field contains cipher text, not plain text.

Notes:

- o It is RECOMMENDED that designers carefully consider the effect of setting flags that delete the bundle in the event that this block cannot be processed.
- o The BCB block processing control flags can be set independently from the processing control flags of the security target(s). The setting of such flags should be an implementation/policy decision for the encrypting node.

3.9. Block Interactions

The security block types defined in this specification are designed to be as independent as possible. However, there are some cases where security blocks may share a security target creating processing dependencies.

If a security target of a BCB is also a security target of a BIB, an undesirable condition occurs where a waypoint would be unable to validate the BIB because one of its security target's contents have been encrypted by a BCB. To address this situation the following processing rules MUST be followed.

- o When adding a BCB to a bundle, if some (or all) of the security targets of the BCB also match all of the security targets of an existing BIB, then the existing BIB MUST also be encrypted. This can be accomplished by either adding a new BCB that targets the existing BIB, or by adding the BIB to the list of security targets for the BCB. Deciding which way to represent this situation is a matter of security policy.

- o When adding a BCB to a bundle, if some (or all) of the security targets of the BCB match some (but not all) of the security targets of a BIB then that BIB MUST be altered in the following way. Any security results in the BIB associated with the BCB security targets MUST be removed from the BIB and placed in a new BIB. This newly created BIB MUST then be encrypted. The encryption of the new BIB can be accomplished by either adding a new BCB that targets the new BIB, or by adding the new BIB to the list of security targets for the BCB. Deciding which way to represent this situation is a matter of security policy.
- o A BIB MUST NOT be added for a security target that is already the security target of a BCB as this would cause ambiguity in block processing order.
- o A BIB integrity value MUST NOT be checked if the BIB is the security target of an existing BCB. In this case, the BIB data is encrypted.
- o A BIB integrity value MUST NOT be checked if the security target associated with that value is also the security target of a BCB. In such a case, the security target data contains cipher text as it has been encrypted.
- o As mentioned in Section 3.7, a BIB MUST NOT have a BCB as its security target.

These restrictions on block interactions impose a necessary ordering when applying security operations within a bundle. Specifically, for a given security target, BIBs MUST be added before BCBs. This ordering MUST be preserved in cases where the current BPA is adding all of the security blocks for the bundle or whether the BPA is a waypoint adding new security blocks to a bundle that already contains security blocks.

In cases where a security source wishes to calculate both a plain text integrity mechanism and encrypt a security target, a BCB with a security context that generates an integrity-protection mechanism as one or more additional security results MUST be used instead of adding both a BIB and then a BCB for the security target at the security source.

3.10. Parameter and Result Identification

Each security context MUST define its own context parameters and results. Each defined parameter and result is represented as the tuple of an identifier and a value. Identifiers are always

represented as a CBOR unsigned integer. The CBOR encoding of values is as defined by the security context specification.

Identifiers MUST be unique for a given security context but do not need to be unique amongst all security contexts.

An example of a security context can be found at [I-D.ietf-dtn-bpsec-default-sc].

3.11. BSP Block Examples

This section provides two examples of BPsec blocks applied to a bundle. In the first example, a single node adds several security operations to a bundle. In the second example, a waypoint node received the bundle created in the first example and adds additional security operations. In both examples, the first column represents blocks within a bundle and the second column represents the Block Number for the block, using the terminology B1...Bn for the purpose of illustration.

3.11.1. Example 1: Constructing a Bundle with Security

In this example a bundle has four non-security-related blocks: the primary block (B1), two extension blocks (B4,B5), and a payload block (B6). The bundle source wishes to provide an integrity signature of the plain text associated with the primary block, the second extension block, and the payload. The bundle source also wishes to provide confidentiality for the first extension block. The resultant bundle is illustrated in Figure 3 and the security actions are described below.

Block in Bundle	ID
Primary Block	B1
BIB OP(bib-integrity, targets=B1, B5, B6)	B2
BCB OP(hcb-confidentiality, target=B4)	B3
Extension Block (encrypted)	B4
Extension Block	B5
Payload Block	B6

Figure 3: Security at Bundle Creation

The following security actions were applied to this bundle at its time of creation.

- o An integrity signature applied to the canonical form of the primary block (B1), the canonical form of the block-type-specific-data field of the second extension block (B5) and the canonical form of the payload block (B6). This is accomplished by a single BIB (B2) with multiple targets. A single BIB is used in this case because all three targets share a security source, security context, and security context parameters. Had this not been the case, multiple BIBs could have been added instead.
- o Confidentiality for the first extension block (B4). This is accomplished by a BCB (B3). Once applied, the block-type-specific-data field of extension block B4 is encrypted. The BCB MUST hold an authentication tag for the cipher text either in the cipher text that now populates the first extension block or as a security result in the BCB itself, depending on which security context is used to form the BCB. A plain text integrity signature may also exist as a security result in the BCB if one is provided by the selected confidentiality security context.

3.11.2. Example 2: Adding More Security At A New Node

Consider that the bundle as it is illustrated in Figure 3 is now received by a waypoint node that wishes to encrypt the second extension block and the bundle payload. The waypoint security policy is to allow existing BIBs for these blocks to persist, as they may be required as part of the security policy at the bundle destination.

The resultant bundle is illustrated in Figure 4 and the security actions are described below. Note that block IDs provided here are ordered solely for the purpose of this example and not meant to impose an ordering for block creation. The ordering of blocks added to a bundle MUST always be in compliance with [I-D.ietf-dtn-bpbis].

Block in Bundle	ID
Primary Block	B1
BIB OP(bib-integrity, targets=B1)	B2
BIB (encrypted) OP(bib-integrity, targets=B5, B6)	B7
BCB OP(bcb-confidentiality, targets=B5, B6, B7)	B8
BCB OP(bcb-confidentiality, target=B4)	B3
Extension Block (encrypted)	B4
Extension Block (encrypted)	B5
Payload Block (encrypted)	B6

Figure 4: Security At Bundle Forwarding

The following security actions were applied to this bundle prior to its forwarding from the waypoint node.

- o Since the waypoint node wishes to encrypt the block-type-specific-data field of blocks B5 and B6, it MUST also encrypt the block-type-specific-data field of the BIBs providing plain text integrity over those blocks. However, BIB B2 could not be encrypted in its entirety because it also held a signature for the primary block (B1). Therefore, a new BIB (B7) is created and security results associated with B5 and B6 are moved out of BIB B2 and into BIB B7.
- o Now that there is no longer confusion of which plain text integrity signatures must be encrypted, a BCB is added to the bundle with the security targets being the second extension block (B5) and the payload (B6) as well as the newly created BIB holding their plain text integrity signatures (B7). A single new BCB is

used in this case because all three targets share a security source, security context, and security context parameters. Had this not been the case, multiple BCBs could have been added instead.

4. Canonical Forms

Security services require consistency and determinism in how information is presented to cipher suites at security sources, verifiers, and acceptors. For example, integrity services require that the same target information (e.g., the same bits in the same order) is provided to the cipher suite when generating an original signature and when validating a signature. Canonicalization algorithms transcode the contents of a security target into a canonical form.

Canonical forms are used to generate input to a security context for security processing at a BP node. If the values of a security target are unchanged, then the canonical form of that target will be the same even if the encoding of those values for wire transmission is different.

BPsec operates on data fields within bundle blocks (e.g., the block-type-specific-data field). In their canonical form, these fields MUST include their own CBOR encoding and MUST NOT include any other encapsulating CBOR encoding. For example, the canonical form of the block-type-specific-data field is a CBOR byte string existing within the CBOR array containing the fields of the extension block. The entire CBOR byte string is considered the canonical block-type-specific-data field. The CBOR array framing is not considered part of the field.

The canonical form of the primary block is as specified in [I-D.ietf-dtn-bpbis] with the following constraint.

- o CBOR values from the primary block MUST be canonicalized using the rules for Deterministically Encoded CBOR, as specified in [RFC8949].

All non-primary blocks share the same block structure and are canonicalized as specified in [I-D.ietf-dtn-bpbis] with the following constraints.

- o CBOR values from the non-primary block MUST be canonicalized using the rules for Deterministically Encoded CBOR, as specified in [RFC8949].

- o Only the block-type-specific-data field may be provided to a cipher suite for encryption as part of a confidentiality security service. Other fields within a non-primary-block MUST NOT be encrypted or decrypted and MUST NOT be included in the canonical form used by the cipher suite for encryption and decryption. These other fields MAY have an integrity protection mechanism applied to them by treating them as associated authenticated data.
- o Reserved and unassigned flags in the block processing control flags field MUST be set to 0 in a canonical form as it is not known if those flags will change in transit.

Security contexts MAY define their own canonicalization algorithms and require the use of those algorithms over the ones provided in this specification. In the event of conflicting canonicalization algorithms, algorithms defined in a security context take precedence over this specification when constructing canonical forms for that security context.

5. Security Processing

This section describes the security aspects of bundle processing.

5.1. Bundles Received from Other Nodes

Security blocks must be processed in a specific order when received by a BP node. The processing order is as follows.

- o When BIBs and BCBs share a security target, BCBs MUST be evaluated first and BIBs second.

5.1.1. Receiving BCBs

If a received bundle contains a BCB, the receiving node MUST determine whether it is the security acceptor for any of the security operations in the BCB. If so, the node MUST process those operations and remove any operation-specific information from the BCB prior to delivering data to an application at the node or forwarding the bundle. If processing a security operation fails, the target SHALL be processed according to the security policy. A bundle status report indicating the failure MAY be generated. When all security operations for a BCB have been removed from the BCB, the BCB MUST be removed from the bundle.

If the receiving node is the destination of the bundle, the node MUST decrypt any BCBs remaining in the bundle. If the receiving node is not the destination of the bundle, the node MUST process the BCB if directed to do so as a matter of security policy.

If the security policy of a node specifies that a node should have applied confidentiality to a specific security target and no such BCB is present in the bundle, then the node MUST process this security target in accordance with the security policy. It is RECOMMENDED that the node remove the security target from the bundle because the confidentiality (and possibly the integrity) of the security target cannot be guaranteed. If the removed security target is the payload block, the bundle MUST be discarded.

If an encrypted payload block cannot be decrypted (i.e., the cipher text cannot be authenticated), then the bundle MUST be discarded and processed no further. If an encrypted security target other than the payload block cannot be decrypted then the associated security target and all security blocks associated with that target MUST be discarded and processed no further. In both cases, requested status reports (see [I-D.ietf-dtn-bpbis]) MAY be generated to reflect bundle or block deletion.

When a BCB is decrypted, the recovered plain text for each security target MUST replace the cipher text in each of the security targets' block-type-specific-data fields. If the plain text is of different size than the cipher text, the CBOR byte string framing of this field must be updated to ensure this field remains a valid CBOR byte string. The length of the recovered plain text is known by the decrypting security context.

If a BCB contains multiple security operations, each operation processed by the node MUST be treated as if the security operation has been represented by a single BCB with a single security operation for the purposes of report generation and policy processing.

5.1.2. Receiving BIBs

If a received bundle contains a BIB, the receiving node MUST determine whether it is the security acceptor for any of the security operations in the BIB. If so, the node MUST process those operations and remove any operation-specific information from the BIB prior to delivering data to an application at the node or forwarding the bundle. If processing a security operation fails, the target SHALL be processed according to the security policy. A bundle status report indicating the failure MAY be generated. When all security operations for a BIB have been removed from the BIB, the BIB MUST be removed from the bundle.

A BIB MUST NOT be processed if the security target of the BIB is also the security target of a BCB in the bundle. Given the order of operations mandated by this specification, when both a BIB and a BCB share a security target, it means that the security target must have

been encrypted after it was integrity signed and, therefore, the BIB cannot be verified until the security target has been decrypted by processing the BCB.

If the security policy of a node specifies that a node should have applied integrity to a specific security target and no such BIB is present in the bundle, then the node MUST process this security target in accordance with the security policy. It is RECOMMENDED that the node remove the security target from the bundle if the security target is not the payload or primary block. If the security target is the payload or primary block, the bundle MAY be discarded. This action can occur at any node that has the ability to verify an integrity signature, not just the bundle destination.

If a receiving node is not the security acceptor of a security operation in a BIB it MAY attempt to verify the security operation anyway to prevent forwarding corrupt data. If the verification fails, the node SHALL process the security target in accordance to local security policy. It is RECOMMENDED that if a payload integrity check fails at a waypoint that it is processed in the same way as if the check fails at the bundle destination. If the check passes, the node MUST NOT remove the security operation from the BIB prior to forwarding.

If a BIB contains multiple security operations, each operation processed by the node MUST be treated as if the security operation has been represented by a single BIB with a single security operation for the purposes of report generation and policy processing.

5.2. Bundle Fragmentation and Reassembly

If it is necessary for a node to fragment a bundle payload, and security services have been applied to that bundle, the fragmentation rules described in [I-D.ietf-dtn-bpbis] MUST be followed. As defined there and summarized here for completeness, only the payload block can be fragmented; security blocks, like all extension blocks, can never be fragmented.

Due to the complexity of payload block fragmentation, including the possibility of fragmenting payload block fragments, integrity and confidentiality operations are not to be applied to a bundle representing a fragment. Specifically, a BCB or BIB MUST NOT be added to a bundle if the "Bundle is a Fragment" flag is set in the Bundle Processing Control Flags field.

Security processing in the presence of payload block fragmentation may be handled by other mechanisms outside of the BPsec protocol or by applying BPsec blocks in coordination with an encapsulation

mechanism. A node should apply any confidentiality protection prior to performing any fragmentation.

6. Key Management

There exist a myriad of ways to establish, communicate, and otherwise manage key information in a DTN. Certain DTN deployments might follow established protocols for key management whereas other DTN deployments might require new and novel approaches. BPSec assumes that key management is handled as a separate part of network management and this specification neither defines nor requires a specific key management strategy.

7. Security Policy Considerations

When implementing BPSec, several policy decisions must be considered. This section describes key policies that affect the generation, forwarding, and receipt of bundles that are secured using this specification. No single set of policy decisions is envisioned to work for all secure DTN deployments.

- o If a bundle is received that contains combinations of security operations that are disallowed by this specification the BPA must determine how to handle the bundle. The bundle may be discarded, the block affected by the security operation may be discarded, or one security operation may be favored over another.
- o BPAs in the network must understand what security operations they should apply to bundles. This decision may be based on the source of the bundle, the destination of the bundle, or some other information related to the bundle.
- o If a waypoint has been configured to add a security operation to a bundle, and the received bundle already has the security operation applied, then the receiver must understand what to do. The receiver may discard the bundle, discard the security target and associated BPsec blocks, replace the security operation, or some other action.
- o It is RECOMMENDED that security operations be applied to every block in a bundle and that the default behavior of a bundle agent is to use the security services defined in this specification. Designers should only deviate from the use of security operations when the deviation can be justified – such as when doing so causes downstream errors when processing blocks whose contents must be inspected or changed at one or more hops along the path.

- o BCB security contexts can alter the size of extension blocks and the payload block. Security policy SHOULD consider how changes to the size of a block could negatively effect bundle processing (e.g., calculating storage needs and scheduling transmission times).
- o Adding a BIB to a security target that has already been encrypted by a BCB is not allowed. If this condition is likely to be encountered, there are (at least) three possible policies that could handle this situation.
 1. At the time of encryption, a security context can be selected which computes a plain text integrity-protection mechanism that is included as a security context result field.
 2. The encrypted block may be replicated as a new block with a new block number and given integrity protection.
 3. An encapsulation scheme may be applied to encapsulate the security target (or the entire bundle) such that the encapsulating structure is, itself, no longer the security target of a BCB and may therefore be the security target of a BIB.
- o Security policy SHOULD address whether cipher suites whose cipher text is larger than the initial plain text are permitted and, if so, for what types of blocks. Changing the size of a block may cause processing difficulties for networks that calculate block offsets into bundles or predict transmission times or storage availability as a function of bundle size. In other cases, changing the size of a payload as part of encryption has no significant impact.

7.1. Security Reason Codes

Bundle protocol agents (BPAs) must process blocks and bundles in accordance with both BP policy and BPsec policy. The decision to receive, forward, deliver, or delete a bundle may be communicated to the report-to address of the bundle, in the form of a status report, as a method of tracking the progress of the bundle through the network. The status report for a bundle may be augmented with a "reason code" explaining why the particular action was taken on the bundle.

This section describes a set of reason codes associated with the security processing of a bundle. The communication of security-related status reports might reduce the security of a network if these reports are intercepted by unintended recipients. BPsec policy

SHOULD specify the conditions in which sending security reason codes are appropriate. Examples of appropriate conditions for the use of security reason codes could include the following.

- o When the report-to address is verified as unchanged from the bundle source. This can occur by placing an appropriate BIB on the bundle primary block.
- o When the block containing a status report with a security reason code is encrypted by a BCB.
- o When a status report containing a security reason code is only sent for security issues relating to bundles and/or blocks associated with non-operational user data or otherwise with test data.
- o When a status report containing a security reason code is only sent for security issues associated with non-operational security contexts, or security contexts using non-operational configurations, such as test keys.

Security reason codes are assigned in accordance with Section 11.2 and are as described below.

Missing Security Operation:

This reason code indicates that a bundle was missing one or more required security operations. This reason code is typically used by a security verifier or security acceptor.

Unknown Security Operation:

This reason code indicates that one or more security operations present in a bundle cannot be understood by the security verifier or security acceptor for the operation. For example, this reason code may be used if a security block references an unknown security context identifier or security context parameter. This reason code should not be used for security operations for which the node is not a security verifier or security acceptor; there is no requirement that all nodes in a network understand all security contexts, security context parameters, and security services for every bundle in a network.

Unexpected Security Operation:

This reason code indicates that a receiving node is neither a security verifier nor a security acceptor for at least one security operation in a bundle. This reason code should not be seen as an error condition; not every node is a security verifier or security acceptor for every security operation in

every bundle. In certain networks, this reason code may be useful in identifying misconfigurations of security policy.

Failed Security Operation:

This reason code indicates that one or more security operations in a bundle failed to process as expected for reasons other than misconfiguration. This may occur when a security-source is unable to add a security block to a bundle. This may occur if the target of a security operation fails to verify using the defined security context at a security verifier. This may also occur if a security operation fails to be processed without error at a security acceptor.

Conflicting Security Operations:

This reason code indicates that two or more security operations in a bundle are not conformant with the BPSec specification and that security processing was unable to proceed because of a BPSec protocol violation.

8. Security Considerations

Given the nature of DTN applications, it is expected that bundles may traverse a variety of environments and devices which each pose unique security risks and requirements on the implementation of security within BPSec. For these reasons, it is important to introduce key threat models and describe the roles and responsibilities of the BPSec protocol in protecting the confidentiality and integrity of the data against those threats. This section provides additional discussion on security threats that BPSec will face and describes how BPSec security mechanisms operate to mitigate these threats.

The threat model described here is assumed to have a set of capabilities identical to those described by the Internet Threat Model in [RFC3552], but the BPSec threat model is scoped to illustrate threats specific to BPSec operating within DTN environments and therefore focuses on on-path-attackers (OPAs). In doing so, it is assumed that the DTN (or significant portions of the DTN) are completely under the control of an attacker.

8.1. Attacker Capabilities and Objectives

BPSec was designed to protect against OPA threats which may have access to a bundle during transit from its source, Alice, to its destination, Bob. An OPA node, Olive, is a non-cooperative node operating on the DTN between Alice and Bob that has the ability to receive bundles, examine bundles, modify bundles, forward bundles, and generate bundles at will in order to compromise the confidentiality or integrity of data within the DTN. There are three

classes of OPA nodes which are differentiated based on their access to cryptographic material:

- o Unprivileged Node: Olive has not been provisioned within the secure environment and only has access to cryptographic material which has been publicly-shared.
- o Legitimate Node: Olive is within the secure environment and therefore has access to cryptographic material which has been provisioned to Olive (i.e., K_M) as well as material which has been publicly-shared.
- o Privileged Node: Olive is a privileged node within the secure environment and therefore has access to cryptographic material which has been provisioned to Olive, Alice and/or Bob (i.e. K_M , K_A , and/or K_B) as well as material which has been publicly-shared.

If Olive is operating as a privileged node, this is tantamount to compromise; BPsec does not provide mechanisms to detect or remove Olive from the DTN or BPsec secure environment. It is up to the BPsec implementer or the underlying cryptographic mechanisms to provide appropriate capabilities if they are needed. It should also be noted that if the implementation of BPsec uses a single set of shared cryptographic material for all nodes, a legitimate node is equivalent to a privileged node because $K_M == K_A == K_B$. For this reason, sharing cryptographic material in this way is not recommended.

A special case of the legitimate node is when Olive is either Alice or Bob (i.e., $K_M == K_A$ or $K_M == K_B$). In this case, Olive is able to impersonate traffic as either Alice or Bob, respectively, which means that traffic to and from that node can be decrypted and encrypted, respectively. Additionally, messages may be signed as originating from one of the endpoints.

8.2. Attacker Behaviors and BPsec Mitigations

8.2.1. Eavesdropping Attacks

Once Olive has received a bundle, she is able to examine the contents of that bundle and attempt to recover any protected data or cryptographic keying material from the blocks contained within. The protection mechanism that BPsec provides against this action is the BCB, which encrypts the contents of its security target, providing confidentiality of the data. Of course, it should be assumed that Olive is able to attempt offline recovery of encrypted data, so the

cryptographic mechanisms selected to protect the data should provide a suitable level of protection.

When evaluating the risk of eavesdropping attacks, it is important to consider the lifetime of bundles on a DTN. Depending on the network, bundles may persist for days or even years. Long-lived bundles imply that the data exists in the network for a longer period of time and, thus, there may be more opportunities to capture those bundles. Additionally, bundles that are long-lived imply that the information stored within them may remain relevant and sensitive for long enough that, once captured, there is sufficient time to crack encryption associated with the bundle. If a bundle does persist on the network for years and the cipher suite used for a BCB provides inadequate protection, Olive may be able to recover the protected data either before that bundle reaches its intended destination or before the information in the bundle is no longer considered sensitive.

NOTE: Olive is not limited by the bundle lifetime and may retain a given bundle indefinitely.

NOTE: Irrespective of whether BPsec is used, traffic analysis will be possible.

8.2.2. Modification Attacks

As a node participating in the DTN between Alice and Bob, Olive will also be able to modify the received bundle, including non-BPsec data such as the primary block, payload blocks, or block processing control flags as defined in [I-D.ietf-dtn-bpbis]. Olive will be able to undertake activities which include modification of data within the blocks, replacement of blocks, addition of blocks, or removal of blocks. Within BPsec, both the BIB and BCB provide integrity protection mechanisms to detect or prevent data manipulation attempts by Olive.

The BIB provides that protection to another block which is its security target. The cryptographic mechanisms used to generate the BIB should be strong against collision attacks and Olive should not have access to the cryptographic material used by the originating node to generate the BIB (e.g., K_A). If both of these conditions are true, Olive will be unable to modify the security target or the BIB and lead Bob to validate the security target as originating from Alice.

Since BPsec security operations are implemented by placing blocks in a bundle, there is no in-band mechanism for detecting or correcting certain cases where Olive removes blocks from a bundle. If Olive removes a BCB, but keeps the security target, the security target

remains encrypted and there is a possibility that there may no longer be sufficient information to decrypt the block at its destination. If Olive removes both a BCB (or BIB) and its security target there is no evidence left in the bundle of the security operation. Similarly, if Olive removes the BIB but not the security target there is no evidence left in the bundle of the security operation. In each of these cases, the implementation of BPSec must be combined with policy configuration at endpoints in the network which describe the expected and required security operations that must be applied on transmission and are expected to be present on receipt. This or other similar out-of-band information is required to correct for removal of security information in the bundle.

A limitation of the BIB may exist within the implementation of BIB validation at the destination node. If Olive is a legitimate node within the DTN, the BIB generated by Alice with K_A can be replaced with a new BIB generated with K_M and forwarded to Bob. If Bob is only validating that the BIB was generated by a legitimate user, Bob will acknowledge the message as originating from Olive instead of Alice. Validating a BIB indicates only that the BIB was generated by a holder of the relevant key; it does not provide any guarantee that the bundle or block was created by the same entity. In order to provide verifiable integrity checks BCB should require an encryption scheme that is Indistinguishable under adaptive Chosen Ciphertext Attack (IND-CCA2) secure. Such an encryption scheme will guard against signature substitution attempts by Olive. In this case, Alice creates a BIB with the protected data block as the security target and then creates a BCB with both the BIB and protected data block as its security targets.

8.2.3. Topology Attacks

If Olive is in a OPA position within the DTN, she is able to influence how any bundles that come to her may pass through the network. Upon receiving and processing a bundle that must be routed elsewhere in the network, Olive has three options as to how to proceed: not forward the bundle, forward the bundle as intended, or forward the bundle to one or more specific nodes within the network.

Attacks that involve re-routing the packets throughout the network are essentially a special case of the modification attacks described in this section where the attacker is modifying fields within the primary block of the bundle. Given that BPSec cannot encrypt the contents of the primary block, alternate methods must be used to prevent this situation. These methods may include requiring BIBs for primary blocks, using encapsulation, or otherwise strategically manipulating primary block data. The specifics of any such

mitigation technique are specific to the implementation of the deploying network and outside of the scope of this document.

Furthermore, routing rules and policies may be useful in enforcing particular traffic flows to prevent topology attacks. While these rules and policies may utilize some features provided by BPsec, their definition is beyond the scope of this specification.

8.2.4. Message Injection

Olive is also able to generate new bundles and transmit them into the DTN at will. These bundles may either be copies or slight modifications of previously-observed bundles (i.e., a replay attack) or entirely new bundles generated based on the Bundle Protocol, BPsec, or other bundle-related protocols. With these attacks Olive's objectives may vary, but may be targeting either the bundle protocol or application-layer protocols conveyed by the bundle protocol. The target could also be the storage and compute of the nodes running the bundle or application layer protocols (e.g., a denial of service to flood on the storage of the store-and-forward mechanism; or compute which would process the packets and perhaps prevent other activities).

BPsec relies on cipher suite capabilities to prevent replay or forged message attacks. A BCB used with appropriate cryptographic mechanisms may provide replay protection under certain circumstances. Alternatively, application data itself may be augmented to include mechanisms to assert data uniqueness and then protected with a BIB, a BCB, or both along with other block data. In such a case, the receiving node would be able to validate the uniqueness of the data.

For example, a BIB may be used to validate the integrity of a bundle's primary block, which includes a timestamp and lifetime for the bundle. If a bundle is replayed outside of its lifetime, then the replay attack will fail as the bundle will be discarded. Similarly, additional blocks such as the Bundle Age may be signed and validated to identify replay attacks. Finally, security context parameters within BIB and BCB blocks may include anti-replay mechanisms such as session identifiers, nonces, and dynamic passwords as supported by network characteristics.

9. Security Context Considerations

9.1. Mandating Security Contexts

Because of the diversity of networking scenarios and node capabilities that may utilize BPsec there is a risk that a single security context mandated for every possible BPsec implementation is

not feasible. For example, a security context appropriate for a resource-constrained node with limited connectivity may be inappropriate for use in a well-resourced, well connected node.

This does not mean that the use of BPsec in a particular network is meant to be used without security contexts for interoperability and default behavior. Network designers must identify the minimal set of security contexts necessary for functions in their network. For example, a default set of security contexts could be created for use over the terrestrial Internet and required by any BPsec implementation communicating over the terrestrial Internet.

To ensure interoperability among various implementations, all BPsec implementations MUST support at least the current IETF standards-track mandatory security context(s). As of this writing, that BCP mandatory security context is specified in [I-D.ietf-dtn-bpsec-default-sc], but the mandatory security context(s) might change over time in accordance with usual IETF processes. Such changes are likely to occur in the future if/when flaws are discovered in the applicable cryptographic algorithms, for example.

Additionally, BPsec implementations need to support the security contexts which are specified and/or used by the BP networks in which they are deployed.

If a node serves as a gateway amongst two or more networks, the BPsec implementation at that node needs to support the union of security contexts mandated in those networks.

BPsec has been designed to allow for a diversity of security contexts and for new contexts to be defined over time. The use of different security contexts does not change the BPsec protocol itself and the definition of new security contexts MUST adhere to the requirements of such contexts as presented in this section and generally in this specification.

Implementors should monitor the state of security context specifications to check for future updates and replacement.

9.2. Identification and Configuration

Security blocks uniquely identify the security context to be used in the processing of their security services. The security context for a security block MUST be uniquely identifiable and MAY use parameters for customization.

To reduce the number of security contexts used in a network, security context designers should make security contexts customizable through the definition of security context parameters. For example, a single security context could be associated with a single cipher suite and security context parameters could be used to configure the use of this security context with different key lengths and different key management options without needing to define separate security contexts for each possible option.

A single security context may be used in the application of more than one security service. This means that a security context identifier MAY be used with a BIB, with a BCB, or with any other BPsec-compliant security block. The definition of a security context MUST identify which security services may be used with the security context, how security context parameters are interpreted as a function of the security operation being supported, and which security results are produced for each security service.

Network operators must determine the number, type, and configuration of security contexts in a system. Networks with rapidly changing configurations may define relatively few security contexts with each context customized with multiple parameters. For networks with more stability, or an increased need for confidentiality, a larger number of contexts can be defined with each context supporting few, if any, parameters.

Security Context Examples

Context Type	Parameters	Definition
Key Exchange AES	Encrypted Key, IV	AES-GCM-256 cipher suite with provided ephemeral key encrypted with a predetermined key encryption key and clear text initialization vector.
Pre-shared Key AES	IV	AES-GCM-256 cipher suite with predetermined key and predetermined key rotation policy.
Out of Band AES	None	AES-GCM-256 cipher suite with all info predetermined.

Table 1

9.3. Authorship

Developers or implementers should consider the diverse performance and conditions of networks on which the Bundle Protocol (and therefore BPsec) will operate. Specifically, the delay and capacity of delay-tolerant networks can vary substantially. Developers should consider these conditions to better describe the conditions when those contexts will operate or exhibit vulnerability, and selection of these contexts for implementation should be made with consideration for this reality. There are key differences that may limit the opportunity for a security context to leverage existing cipher suites and technologies that have been developed for use in traditional, more reliable networks:

- o **Data Lifetime:** Depending on the application environment, bundles may persist on the network for extended periods of time, perhaps even years. Cryptographic algorithms should be selected to ensure protection of data against attacks for a length of time reasonable for the application.
- o **One-Way Traffic:** Depending on the application environment, it is possible that only a one-way connection may exist between two endpoints, or if a two-way connection does exist, the round-trip time may be extremely large. This may limit the utility of session key generation mechanisms, such as Diffie-Hellman, as a two-way handshake may not be feasible or reliable.
- o **Opportunistic Access:** Depending on the application environment, a given endpoint may not be guaranteed to be accessible within a certain amount of time. This may make asymmetric cryptographic architectures which rely on a key distribution center or other trust center impractical under certain conditions.

When developing security contexts for use with BPsec, the following information SHOULD be considered for inclusion in these specifications.

- o **Security Context Parameters.** Security contexts MUST define their parameter Ids, the data types of those parameters, and their CBOR encoding.
- o **Security Results.** Security contexts MUST define their security result Ids, the data types of those results, and their CBOR encoding.
- o **New Canonicalizations.** Security contexts may define new canonicalization algorithms as necessary.

- o Cipher-Text Size. Security contexts MUST state whether their associated cipher suites generate cipher text (to include any authentication information) that is of a different size than the input plain text.

If a security context does not wish to alter the size of the plain text it should place overflow bytes and authentication tags in security result fields.

- o Block Header Information. Security contexts SHOULD include block header information that is considered to be immutable for the block. This information MAY include the block type code, block number, CRC Type and CRC field (if present or if missing and unlikely to be added later), and possibly certain block processing control flags. Designers should input these fields as additional data for integrity protection when these fields are expected to remain unchanged over the path the block will take from the security source to the security acceptor. Security contexts considering block header information MUST describe expected behavior when these fields fail their integrity verification.
- o Handling CRC Fields. Security contexts may include algorithms that alter the contexts of their security target block, such as the case when encrypting the block-type-specific data of a target block as part of a BCB confidentiality service. Security context specifications SHOULD address how preexisting CRC-Type and CRC-Value fields be handled. For example, a BCB security context could remove the plain-text CRC value from its target upon encryption and replace or recalculate the value upon decryption.

10. Defining Other Security Blocks

Other security blocks (OSBs) may be defined and used in addition to the security blocks identified in this specification. Both the usage of BIB, BCB, and any future OSBs can co-exist within a bundle and can be considered in conformance with BPsec if each of the following requirements are met by any future identified security blocks.

- o Other security blocks (OSBs) MUST NOT reuse any enumerations identified in this specification, to include the block type codes for BIB and BCB.
- o An OSB definition MUST state whether it can be the target of a BIB or a BCB. The definition MUST also state whether the OSB can target a BIB or a BCB.
- o An OSB definition MUST provide a deterministic processing order in the event that a bundle is received containing BIBs, BCBs, and

OSBs. This processing order **MUST NOT** alter the BIB and BCB processing orders identified in this specification.

- o An OSB definition **MUST** provide a canonicalization algorithm if the default non-primary-block canonicalization algorithm cannot be used to generate a deterministic input for a cipher suite. This requirement can be waived if the OSB is defined so as to never be the security target of a BIB or a BCB.
- o An OSB definition **MUST NOT** require any behavior of a BPSEC-BPA that is in conflict with the behavior identified in this specification. In particular, the security processing requirements imposed by this specification must be consistent across all BPSEC-BPAs in a network.
- o The behavior of an OSB when dealing with fragmentation must be specified and **MUST NOT** lead to ambiguous processing states. In particular, an OSB definition should address how to receive and process an OSB in a bundle fragment that may or may not also contain its security target. An OSB definition should also address whether an OSB may be added to a bundle marked as a fragment.

Additionally, policy considerations for the management, monitoring, and configuration associated with blocks **SHOULD** be included in any OSB definition.

NOTE: The burden of showing compliance with processing rules is placed upon the specifications defining new security blocks and the identification of such blocks shall not, alone, require maintenance of this specification.

11. IANA Considerations

This specification includes fields requiring registries managed by IANA.

11.1. Bundle Block Types

This specification allocates two block types from the existing "Bundle Block Types" registry defined in [RFC6255].

Additional Entries for the Bundle Block-Type Codes Registry:

Value	Description	Reference
TBA	Block Integrity Block	This document
TBA	Block Confidentiality Block	This document

Table 2

The Bundle Block Types namespace notes whether a block type is meant for use in BP version 6, BP version 7, or both. The two block types defined in this specification are meant for use with BP version 7.

11.2. Bundle Status Report Reason Codes

This specification allocates five reason codes from the existing "Bundle Status Report Reason Codes" registry defined in [RFC6255].

Additional Entries for the Bundle Status Report Reason Codes Registry:

BP Version	Value	Description	Reference
7	TBD	Missing Security Operation	This document, Section 7.1
7	TBD	Unknown Security Operation	This document, Section 7.1
7	TBD	Unexpected Security Operation	This document, Section 7.1
7	TBD	Failed Security Operation	This document, Section 7.1
7	TBD	Conflicting Security Operation	This document, Section 7.1

11.3. Security Context Identifiers

BPsec has a Security Context Identifier field for which IANA is requested to create and maintain a new registry named "BPsec Security Context Identifiers". Initial values for this registry are given below.

The registration policy for this registry is: Specification Required.

The value range is: signed 16-bit integer.

BPsec Security Context Identifier Registry

Value	Description	Reference
< 0	Reserved	This document
0	Reserved	This document

Table 3

Negative security context identifiers are reserved for local/site-specific uses. The use of 0 as a security context identifier is for non-operational testing purposes only.

12. References

12.1. Normative References

[I-D.ietf-dtn-bpbis]

Burleigh, S., Fall, K., and E. Birrane, "Bundle Protocol Version 7", draft-ietf-dtn-bpbis-31 (work in progress), January 2021.

[I-D.ietf-dtn-bpsec-default-sc]

Birrane, E., "BPsec Default Security Contexts", draft-ietf-dtn-bpsec-default-sc-01 (work in progress), February 2021.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<https://www.rfc-editor.org/info/rfc3552>>.

[RFC6255] Blanchet, M., "Delay-Tolerant Networking Bundle Protocol IANA Registries", RFC 6255, DOI 10.17487/RFC6255, May 2011, <<https://www.rfc-editor.org/info/rfc6255>>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/info/rfc8949>>.

12.2. Informative References

- [I-D.birrane-dtn-sbsp]
 Birrane, E., Pierce-Mayer, J., and D. Iannicca,
 "Streamlined Bundle Security Protocol Specification",
 draft-birrane-dtn-sbsp-01 (work in progress), October 2015.
- [RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <<https://www.rfc-editor.org/info/rfc4838>>.
- [RFC6257] Symington, S., Farrell, S., Weiss, H., and P. Lovell, "Bundle Security Protocol Specification", RFC 6257, DOI 10.17487/RFC6257, May 2011, <<https://www.rfc-editor.org/info/rfc6257>>.

Appendix A. Acknowledgements

The following participants contributed technical material, use cases, and useful thoughts on the overall approach to this security specification: Scott Burleigh of the Jet Propulsion Laboratory, Angela Hennessy of the Laboratory for Telecommunications Sciences, and Amy Alford, Angela Dalton, and Cherita Corbett of the Johns Hopkins University Applied Physics Laboratory.

Authors' Addresses

Edward J. Birrane, III
The Johns Hopkins University Applied
 Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
US

Phone: +1 443 778 7423
Email: Edward.Birrane@jhuapl.edu

Kenneth McKeever
The Johns Hopkins University Applied
 Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
US

Phone: +1 443 778 2237
Email: Ken.McKeever@jhuapl.edu

Delay Tolerant Networking
Internet-Draft
Obsoletes: RFC7242 (if approved)
Intended status: Standards Track
Expires: May 31, 2017

B. Sipos
RKF Engineering
M. Demmer
UC Berkeley
J. Ott
Aalto University
S. Perreault
November 27, 2016

Delay-Tolerant Networking TCP Convergence Layer Protocol Version 4
draft-ietf-dtn-tcpclv4-01

Abstract

This document describes a revised protocol for the TCP-based convergence layer for Delay-Tolerant Networking (DTN). The protocol revision is based on implementation issues in the original [RFC7242] and updates to the Bundle Protocol contents, encodings, and convergence layer requirements in [I-D.ietf-dtn-bpbis].

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 31, 2017.

Copyright Notice

Copyright (c) 2016 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 Simplified BSD License.

Table of Contents

1. Introduction	3
2. Requirements Language	4
2.1. Definitions Specific to the TCPCL Protocol	4
3. General Protocol Description	5
3.1. Bidirectional Use of TCPCL Sessions	6
3.2. Example Message Exchange	6
4. Session Establishment	7
4.1. Contact Header	8
4.2. Validation and Parameter Negotiation	10
5. Established Session Operation	11
5.1. Message Type Codes	11
5.2. Upkeep and Status Messages	12
5.2.1. Session Upkeep (KEEPALIVE)	12
5.2.2. Message Rejection (REJECT)	13
5.3. Session Security	14
5.3.1. TLS Handshake Result	14
5.3.2. Example TLS Initiation	15
5.4. Bundle Transfer	15
5.4.1. Bundle Transfer ID	16
5.4.2. Bundle Length (LENGTH)	16
5.4.3. Bundle Data Transmission (DATA_SEGMENT)	17
5.4.4. Bundle Acknowledgments (ACK_SEGMENT)	18
5.4.5. Bundle Refusal (REFUSE_BUNDLE)	19
6. Session Termination	21
6.1. Shutdown Message (SHUTDOWN)	21
6.2. Idle Session Shutdown	23
7. Security Considerations	23
8. IANA Considerations	24
8.1. Port Number	25
8.2. Protocol Versions	25
8.3. Message Types	26
8.4. REFUSE_BUNDLE Reason Codes	26
8.5. SHUTDOWN Reason Codes	27
8.6. REJECT Reason Codes	27
9. Acknowledgments	28
10. References	28
10.1. Normative References	28
10.2. Informative References	29
Appendix A. Significant changes from RFC7242	29
Authors' Addresses	30

1. Introduction

This document describes the TCP-based convergence-layer protocol for Delay-Tolerant Networking. Delay-Tolerant Networking is an end-to-end architecture providing communications in and/or through highly stressed environments, including those with intermittent connectivity, long and/or variable delays, and high bit error rates. More detailed descriptions of the rationale and capabilities of these networks can be found in "Delay-Tolerant Network Architecture" [RFC4838].

An important goal of the DTN architecture is to accommodate a wide range of networking technologies and environments. The protocol used for DTN communications is the revised Bundle Protocol (BP) [I-D.ietf-dtn-bpbis], an application-layer protocol that is used to construct a store-and-forward overlay network. As described in the Bundle Protocol specification [I-D.ietf-dtn-bpbis], it requires the services of a "convergence-layer adapter" (CLA) to send and receive bundles using the service of some "native" link, network, or Internet protocol. This document describes one such convergence-layer adapter that uses the well-known Transmission Control Protocol (TCP). This convergence layer is referred to as TCPCL.

The locations of the TCPCL and the BP in the Internet model protocol stack are shown in Figure 1. In particular, when BP is using TCP as its bearer with TCPCL as its convergence layer, both BP and TCPCL reside at the application layer of the Internet model.

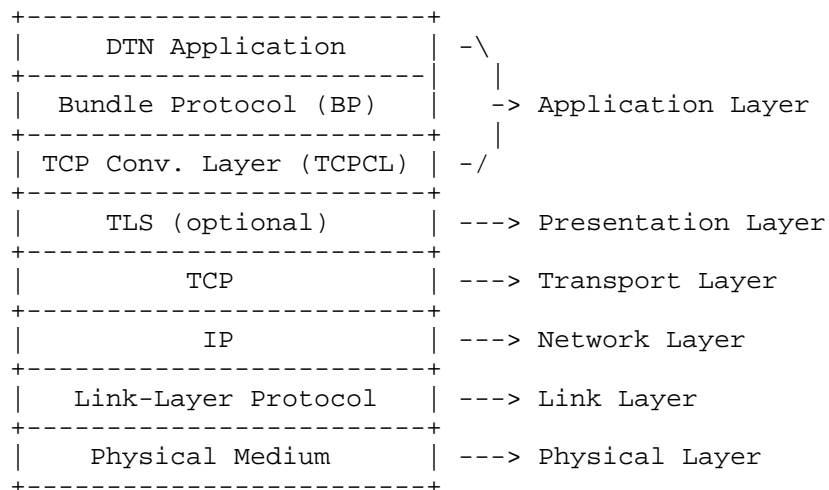


Figure 1: The Locations of the Bundle Protocol and the TCP Convergence-Layer Protocol above the Internet Protocol Stack

This document describes the format of the protocol data units passed between entities participating in TCPCL communications. This document does not address:

- o The format of protocol data units of the Bundle Protocol, as those are defined elsewhere in [RFC5050] and [I-D.ietf-dtn-bpbis]. This includes the concept of bundle fragmentation or bundle encapsulation. The TCPCL transfers bundles as opaque data blocks.
- o Mechanisms for locating or identifying other bundle nodes within an internet.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.1. Definitions Specific to the TCPCL Protocol

This section contains definitions that are interpreted to be specific to the operation of the TCPCL protocol, as described below.

TCP Connection: A TCP connection refers to a transport connection using TCP as the transport protocol.

TCPCL Session: A TCPCL session (as opposed to a TCP connection) is a TCPCL communication relationship between two bundle nodes. The lifetime of a TCPCL session is bound to the lifetime of an underlying TCP connection. Therefore, a TCPCL session is initiated when a bundle node initiates a TCP connection to be established for the purposes of bundle communication. A TCPCL session is terminated when the TCP connection ends, due either to one or both nodes actively terminating the TCP connection or due to network errors causing a failure of the TCP connection. For the remainder of this document, the term "session" without the prefix "TCPCL" refer to a TCPCL session.

Session parameters: The session parameters are a set of values used to affect the operation of the TCPCL for a given session. The manner in which these parameters are conveyed to the bundle node and thereby to the TCPCL is implementation dependent. However, the mechanism by which two bundle nodes exchange and negotiate the values to be used for a given session is described in Section 4.2.

Transmission: Transmission refers to the procedures and mechanisms (described below) for conveyance of a bundle from one node to another.

3. General Protocol Description

The service of this protocol is the transmission of DTN bundles over TCP. This document specifies the encapsulation of bundles, procedures for TCP setup and teardown, and a set of messages and node requirements. The general operation of the protocol is as follows.

First, one node establishes a TCPCL session to the other by initiating a TCP connection. After setup of the TCP connection is complete, an initial contact header is exchanged in both directions to set parameters of the TCPCL session and exchange a singleton endpoint identifier for each node (not the singleton Endpoint Identifier (EID) of any application running on the node) to denote the bundle-layer identity of each DTN node. This is used to assist in routing and forwarding messages, e.g., to prevent loops.

Once the TCPCL session is established and configured in this way, bundles can be transferred in either direction. Each transfer is performed in one or more logical segments of data. Each logical data segment consists of a DATA_SEGMENT message header, a count of the length of the segment, and finally the octet range of the bundle data. The choice of the length to use for segments is an implementation matter (but must be within the Segment MRU size of Section 4.1). The first segment for a bundle MUST set the 'start' flag, and the last one MUST set the 'end' flag in the DATA_SEGMENT message header.

If multiple bundles are transmitted on a single TCPCL connection, they MUST be transmitted consecutively. Interleaving data segments from different bundles is not allowed. Bundle interleaving can be accomplished by fragmentation at the BP layer or by establishing multiple TCPCL sessions.

A feature of this protocol is for the receiving node to send acknowledgments as bundle data segments arrive (ACK_SEGMENT). The rationale behind these acknowledgments is to enable the sender node to determine how much of the bundle has been received, so that in case the session is interrupted, it can perform reactive fragmentation to avoid re-sending the already transmitted part of the bundle. For each data segment that is received, the receiving node sends an ACK_SEGMENT code followed by a count containing the cumulative length of the bundle that has been received. The sending node MAY transmit multiple DATA_SEGMENT messages without necessarily waiting for the corresponding ACK_SEGMENT responses. This enables pipelining of messages on a channel. In addition, there is no explicit flow control on the TCPCL layer.

Another feature is that a receiver MAY interrupt the transmission of a bundle at any point in time by replying with a REFUSE_BUNDLE message, which causes the sender to stop transmission of the current bundle, after completing transmission of a partially sent data segment. Note: This enables a cross-layer optimization in that it allows a receiver that detects that it already has received a certain bundle to interrupt transmission as early as possible and thus save transmission capacity for other bundles.

For sessions that are idle, a KEEPALIVE message is sent at a negotiated interval. This is used to convey liveness information.

Finally, before sessions close, a SHUTDOWN message is sent to the session peer. After sending a SHUTDOWN message, the sender of this message MAY send further acknowledgments (ACK_SEGMENT or REFUSE_BUNDLE) but no further data messages (DATA_SEGMENT). A SHUTDOWN message MAY also be used to refuse a session setup by a peer.

3.1. Bidirectional Use of TCPCL Sessions

There are specific messages for sending and receiving operations (in addition to session setup/teardown). TCPCL is symmetric, i.e., both sides can start sending data segments in a session, and one side's bundle transfer does not have to complete before the other side can start sending data segments on its own. Hence, the protocol allows for a bi-directional mode of communication.

Note that in the case of concurrent bidirectional transmission, acknowledgment segments MAY be interleaved with data segments.

3.2. Example Message Exchange

The following figure visually depicts the protocol exchange for a simple session, showing the session establishment and the transmission of a single bundle split into three data segments (of lengths L1, L2, and L3) from Node A to Node B.

Note that the sending node MAY transmit multiple DATA_SEGMENT messages without necessarily waiting for the corresponding ACK_SEGMENT responses. This enables pipelining of messages on a channel. Although this example only demonstrates a single bundle transmission, it is also possible to pipeline multiple DATA_SEGMENT messages for different bundles without necessarily waiting for ACK_SEGMENT messages to be returned for each one. However, interleaving data segments from different bundles is not allowed.

No errors or rejections are shown in this example.

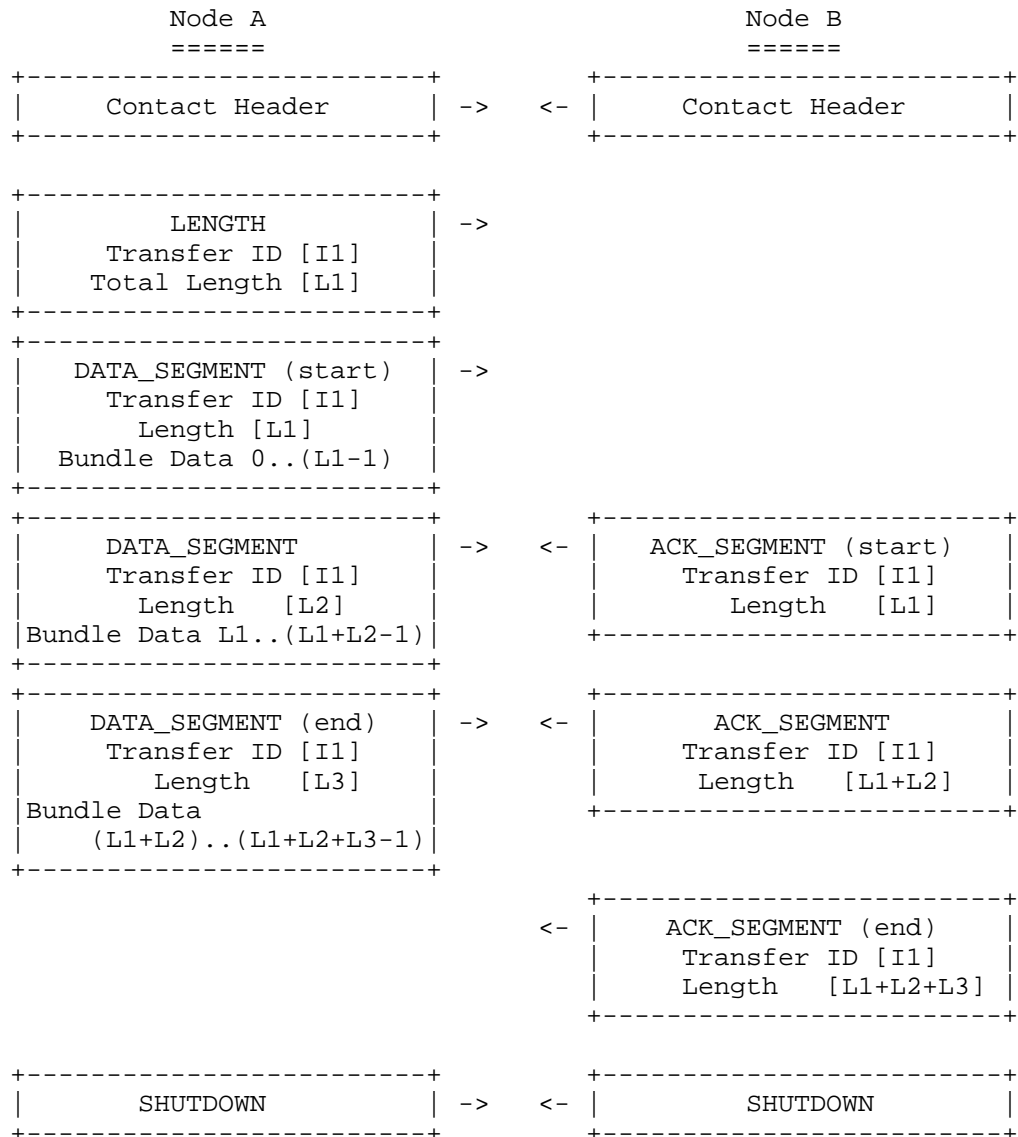


Figure 2: A Simple Visual Example of the Flow of Protocol Messages on a Single TCP Session between Two Nodes (A and B)

4. Session Establishment

For bundle transmissions to occur using the TCPCL, a TCPCL session **MUST** first be established between communicating nodes. It is up to the implementation to decide how and when session setup is triggered.

For example, some sessions MAY be opened proactively and maintained for as long as is possible given the network conditions, while other sessions MAY be opened only when there is a bundle that is queued for transmission and the routing algorithm selects a certain next-hop node.

To establish a TCPCL session, a node MUST first establish a TCP connection with the intended peer node, typically by using the services provided by the operating system. Port number 4556 has been assigned by IANA as the well-known port number for the TCP convergence layer. Other port numbers MAY be used per local configuration. Determining a peer's port number (if different from the well-known TCPCL port) is up to the implementation.

If the node is unable to establish a TCP connection for any reason, then it is an implementation matter to determine how to handle the connection failure. A node MAY decide to re-attempt to establish the connection. If it does so, it MUST NOT overwhelm its target with repeated connection attempts. Therefore, the node MUST retry the connection setup only after some delay (a 1-second minimum is RECOMMENDED), and it SHOULD use a (binary) exponential backoff mechanism to increase this delay in case of repeated failures. In case a SHUTDOWN message specifying a reconnection delay is received, that delay is used as the initial delay. The default initial delay SHOULD be at least 1 second but SHOULD be configurable since it will be application and network type dependent.

The node MAY declare failure after one or more connection attempts and MAY attempt to find an alternate route for bundle data. Such decisions are up to the higher layer (i.e., the BP).

Once a TCP connection is established, each node MUST immediately transmit a contact header over the TCP connection. The format of the contact header is described in Section 4.1.

Upon receipt of the contact header, both nodes perform the validation and negotiation procedures defined in Section 4.2

After receiving the contact header from the other node, either node MAY also refuse the session by sending a SHUTDOWN message. If session setup is refused, a reason MUST be included in the SHUTDOWN message.

4.1. Contact Header

Once a TCP connection is established, both parties exchange a contact header. This section describes the format of the contact header and the meaning of its fields.

The format for the Contact Header is as follows:

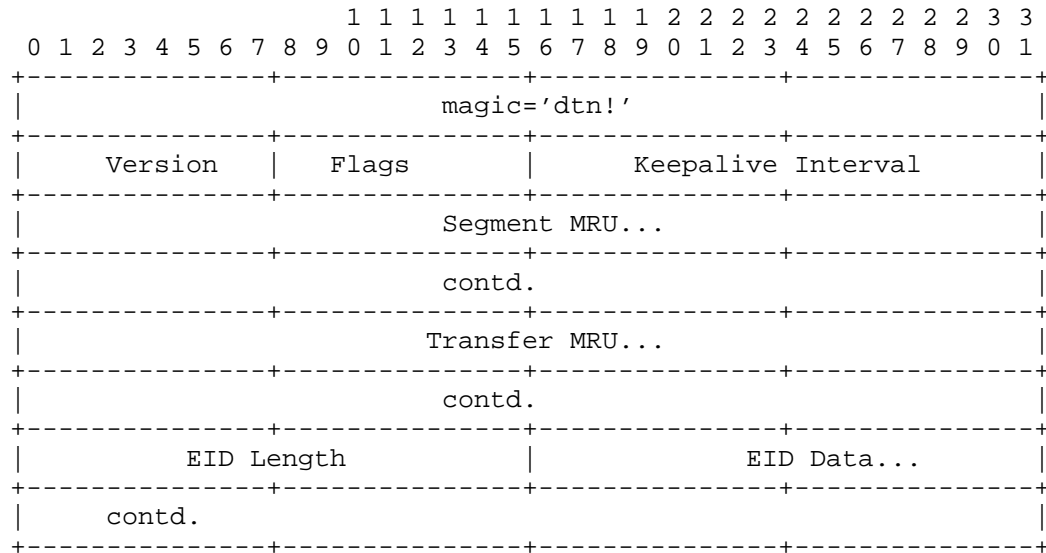


Figure 3: Contact Header Format

The fields of the contact header are:

magic: A four-octet field that always contains the octet sequence 0x64 0x74 0x6e 0x21, i.e., the text string "dtn!" in US-ASCII (and UTF-8).

Version: A one-octet field value containing the value 4 (current version of the protocol).

Flags: A one-octet field of single-bit flags, interpreted according to the descriptions in Table 1.

Keepalive Interval: A 16-bit unsigned integer indicating the longest allowable interval in seconds between KEEPALIVE messages received in this session.

Segment MRU: A 64-bit unsigned integer indicating the largest allowable single-segment data payload size to be received in this session. Any DATA_SEGMENT sent to this peer SHALL have a data payload no longer than the peer's Segment MRU. The two endpoints of a single session MAY have different Segment MRUs, and no relation between the two is required.

Transfer MRU: A 64-bit unsigned integer indicating the largest allowable total-bundle data size to be received in this session. Any bundle transfer sent to this peer SHALL have a Total bundle data payload no longer than the peer's Transfer MRU. This value can be used to perform proactive bundle fragmentation. The two endpoints of a single session MAY have different Transfer MRUs, and no relation between the two is required.

EID Length and EID Data: Together these fields represent a variable-length text string. The EID Length is a 16-bit unsigned integer indicating the number of octets of EID Data to follow. A zero EID Length is a special case which indicates the lack of EID rather than a truly empty EID. A non-zero-length EID Data contains the UTF-8 encoded EID of some singleton endpoint in which the sending node is a member, in the canonical format of <scheme name>:<scheme-specific part>.

Type	Code	Description
CAN_TLS	0x01	If bit is set, indicates that the sending peer is capable of TLS security.

Table 1: Contact Header Flags

4.2. Validation and Parameter Negotiation

Upon reception of the contact header, each node follows the following procedures to ensure the validity of the TCPCL session and to negotiate values for the session parameters.

If the magic string is not present or is not valid, the connection MUST be terminated. The intent of the magic string is to provide some protection against an inadvertent TCP connection by a different protocol than the one described in this document. To prevent a flood of repeated connections from a misconfigured application, a node MAY elect to hold an invalid connection open and idle for some time before closing it.

If a node receives a contact header containing a version that is greater than the current version of the protocol that the node implements, then the node SHALL shutdown the session with a reason code of "Version mismatch". If a node receives a contact header with a version that is lower than the version of the protocol that the node implements, the node MAY either terminate the session (with a reason code of "Version mismatch"). Otherwise, the node MAY adapt

its operation to conform to the older version of the protocol. This decision is an implementation matter.

A node calculates the parameters for a TCPCL session by negotiating the values from its own preferences (conveyed by the contact header it sent to the peer) with the preferences of the peer node (expressed in the contact header that it received from the peer). The negotiated parameters defined by this specification are described in the following paragraphs.

Session Keepalive: Negotiation of the Session Keepalive parameter is performed by taking the minimum of this two contact headers' Keepalive Interval. If the negotiated Session Keepalive is zero (i.e. one or both contact headers contains a zero Keepalive Interval), then the keepalive feature (described in Section 5.2.1) is disabled.

Enable TLS: Negotiation of the Enable TLS parameter is performed by taking the logical AND of the two contact headers' CAN_TLS flags. If the negotiated Enable TLS value is true then TLS negotiation feature (described in Section 5.3) begins immediately following the contact header exchange.

Once this process of parameter negotiation is completed, the protocol defines no additional mechanism to change the parameters of an established session; to effect such a change, the session MUST be terminated and a new session established.

5. Established Session Operation

This section describes the protocol operation for the duration of an established session, including the mechanism for transmitting bundles over the session.

5.1. Message Type Codes

After the initial exchange of a contact header, all messages transmitted over the session are identified by a one-octet header with the following structure:

```

  0 1 2 3 4 5 6 7
+---+---+---+---+---+
| type | flags |
+---+---+---+---+---+
```

Figure 4: Format of the One-Octet Message Header

type: Indicates the type of the message as per Table 2 below.

flags: Optional flags defined based on message type.

The types and values for the message type code are as follows.

Type	Code	Description
DATA_SEGMENT	0x1	Indicates the transmission of a segment of bundle data, as described in Section 5.4.3.
ACK_SEGMENT	0x2	Acknowledges reception of a data segment, as described in Section 5.4.4.
REFUSE_BUNDLE	0x3	Indicates that the transmission of the current bundle SHALL be stopped, as described in Section 5.4.5.
KEEPALIVE	0x4	KEEPALIVE message for the session, as described in Section 5.2.1.
SHUTDOWN	0x5	Indicates that one of the nodes participating in the session wishes to cleanly terminate the session, as described in Section 6.
LENGTH	0x6	Contains the length (in octets) of the next bundle, as described in Section 5.4.2.
REJECT	TBD	Contains a TCPCL message rejection, as described in Section 5.2.2.

Table 2: TCPCL Message Types

5.2. Upkeep and Status Messages

5.2.1. Session Upkeep (KEEPALIVE)

The protocol includes a provision for transmission of KEEPALIVE messages over the TCPCL session to help determine if the underlying TCP connection has been disrupted.

As described in Section 4.1, one of the parameters in the contact header is the Keepalive Interval. Both sides populate this field with their requested intervals (in seconds) between KEEPALIVE messages.

The format of a KEEPALIVE message is a one-octet message type code of KEEPALIVE (as described in Table 2) with no additional data. Both sides SHOULD send a KEEPALIVE message whenever the negotiated interval has elapsed with no transmission of any message (KEEPALIVE or other).

If no message (KEEPALIVE or other) has been received for at least twice the Keepalive Interval, then either party MAY terminate the session by transmitting a one-octet SHUTDOWN message (as described in Table 2, with reason code "Idle Timeout") and by closing the session.

Note: The Keepalive Interval SHOULD not be chosen too short as TCP retransmissions MAY occur in case of packet loss. Those will have to be triggered by a timeout (TCP retransmission timeout (RTO)), which is dependent on the measured RTT for the TCP connection so that KEEPALIVE messages MAY experience noticeable latency.

5.2.2. Message Rejection (REJECT)

If a TCPCL endpoint receives a message which is unknown to it (possibly due to an unhandled protocol mismatch) or is inappropriate for the current session state (e.g. a KEEPALIVE message received after contact header negotiation has disabled that feature), there is a protocol-level message to signal this condition in the form of a REJECT reply.

The format of a REJECT message follows:

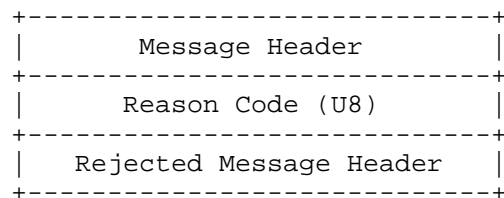


Figure 5: Format of REJECT Messages

The Rejected Message Header is a copy of the Message Header to which the REJECT message is sent as a response. The REJECT Reason Code is an 8-bit unsigned integer and indicates why the REJECT itself was sent. The specified values of the reason code are:

Name	Code	Description
Message Type Unknown	0x01	A message was received with a Message Type code unknown to the TCPCL endpoint.
Message Unsupported	0x02	A message was received but the TCPCL endpoint cannot comply with the message contents.
Message Unexpected	0x03	A message was received while the session is in a state in which the message is not expected.

Table 3: REJECT Reason Codes

5.3. Session Security

This version of the TCPCL supports establishing a session-level Transport Layer Security (TLS) session within an existing TCPCL session. Negotiation of whether or not to initiate TLS within TCPCL session is part of the contact header as described in Section 4.2.

When TLS is used within the TCPCL it affects the entire session. By convention, this protocol uses the endpoint which initiated the underlying TCP connection as the "client" role of the TLS handshake request. Once a TLS session is established within TCPCL, there is no mechanism provided to end the TLS session and downgrade the session. If a non-TLS session is desired after a TLS session is started then the entire TCPCL session MUST be shutdown first.

After negotiating an Enable TLS parameter of true, and before any other TCPCL messages are sent within the session, the session endpoints SHALL begin a TLS handshake in accordance with [RFC5246]. The parameters within each TLS negotiation are implementation dependent but any TCPCL endpoint SHOULD follow all recommended best practices of [RFC7525].

5.3.1. TLS Handshake Result

If a TLS handshake cannot negotiate a TLS session, both endpoints of the TCPCL session SHALL cause a TCPCL shutdown with reason "TLS negotiation failed".

After a TLS session is successfully established, both TCPCL endpoints SHALL re-exchange TCPCL Contact Header messages. Any information

cached from the prior Contact Header exchange SHALL be discarded. This re-exchange avoids man-in-the-middle attack in identical fashion to [RFC2595].

5.3.2. Example TLS Initiation

A summary of a typical CAN_TLS usage is shown in the sequence in Figure 6 below.

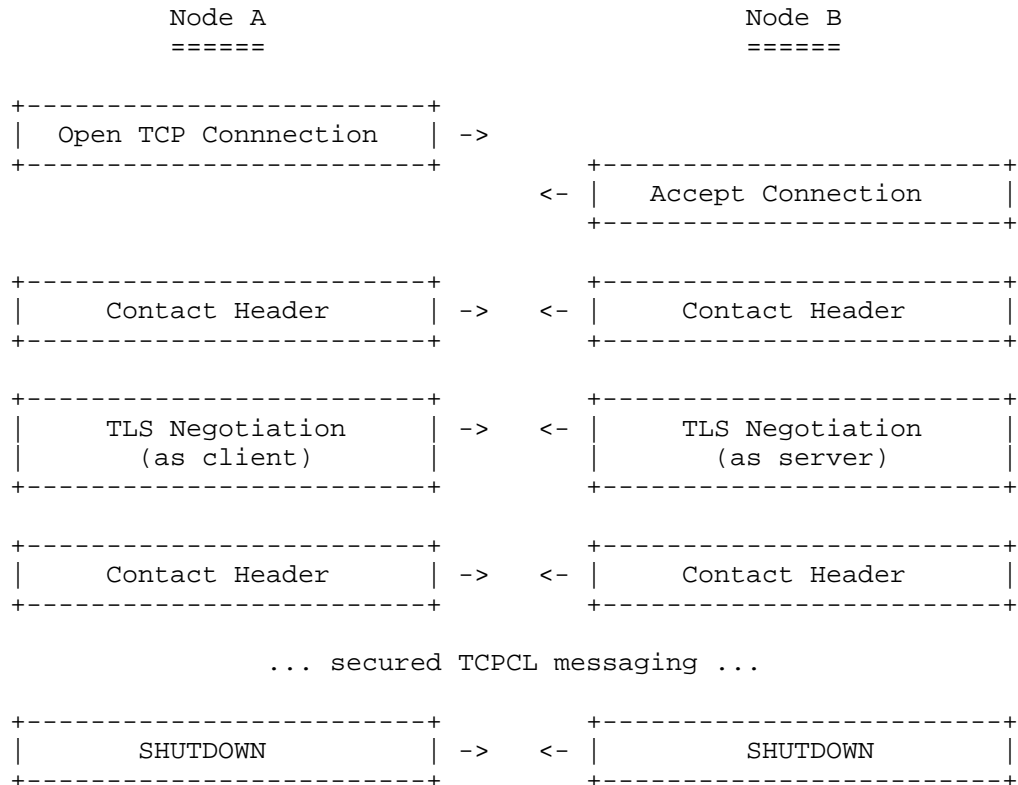


Figure 6: A simple visual example of TCPCL TLS Establishment between two nodes

5.4. Bundle Transfer

All of the message in this section are directly associated with tranfering a bundle between TCPCL endpoints.

A single TCPCL transfer results in a bundle (handled by the convergence layer as opaque data) being exchanged from one endpoint to the other. In TCPCL a transfer is accomplished by dividing a

single bundle up into "segments" based on the receiving-side Segment MRU (see Section 4.1).

A single transfer (and by extension a single segment) SHALL NOT contain data of more than a single bundle. This requirement is imposed on the agent using the TCPCL rather than TCPCL itself.

5.4.1. Bundle Transfer ID

Each of the bundle transfer messages contains a Transfer ID number which is used to correlate messages originating from sender and receiver of a bundle. A Transfer ID does not attempt to address uniqueness of the bundle data itself and has no relation to concepts such as bundle fragmentation. Each invocation of TCPCL by the bundle protocol agent, requesting transmission of a bundle (fragmentary or otherwise), results in the initiation of a single TCPCL transfer. Each transfer entails the sending of a LENGTH message and some number of DATA_SEGMENT and ACK_SEGMENT messages; all are correlated by the same Transfer ID.

Transfer IDs from each endpoint SHALL be unique within a single TCPCL session. The initial Transfer ID from each endpoint SHALL have value zero. Subsequent Transfer ID values SHALL be incremented from the prior Transfer ID value by one. Upon exhaustion of the entire 64-bit Transfer ID space, the sending endpoint SHALL terminate the session with SHUTDOWN reason code "Resource Exhaustion".

For bidirectional bundle transfers, a TCPCL endpoint SHOULD NOT rely on any relation between Transfer IDs originating from each side of the TCPCL session.

5.4.2. Bundle Length (LENGTH)

The LENGTH message contains the total length, in octets, of the bundle data in the associated transfer. The total length is formatted as a 64-bit unsigned integer.

The purpose of the LENGTH message is to allow nodes to preemptively refuse bundles that would exceed their resources or to prepare storage on the receiving node for the upcoming bundle data. See Section 5.4.5 for details on when refusal based on LENGTH content is acceptable.

The Total Bundle Length field within a LENGTH message SHALL be used as informative data by the receiver. If, for whatever reason, the actual total length of bundle data received differs from the value indicated by the LENGTH message, the receiver SHOULD accept the full set of bundle data as valid.

The format of the LENGTH message is as follows:

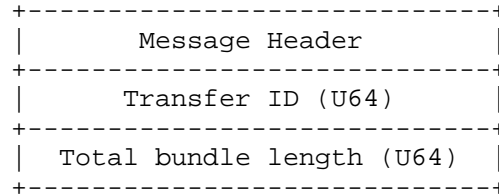


Figure 7: Format of LENGTH Messages

LENGTH messages SHALL be sent immediately before transmission of any DATA_SEGMENT messages. LENGTH messages MUST NOT be sent unless the next DATA_SEGMENT message has the 'S' bit set to "1" (i.e., just before the start of a new transfer).

A receiver MAY send a BUNDLE_REFUSE message as soon as it receives a LENGTH message without waiting for the next DATA_SEGMENT message. The sender MUST be prepared for this and MUST associate the refusal with the correct bundle via the Transfer ID fields.

Upon reception of a LENGTH message not immediately before the start of a starting DATA_SEGMENT the receiver SHALL send a REJECT message with a Reason Code of "Message Unexpected".

5.4.3. Bundle Data Transmission (DATA_SEGMENT)

Each bundle is transmitted in one or more data segments. The format of a DATA_SEGMENT message follows in Figure 8 and its use of header flags is shown in Figure 9.

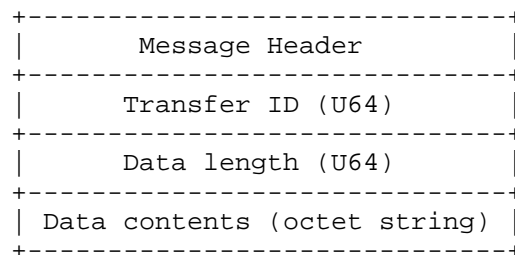


Figure 8: Format of DATA_SEGMENT Messages

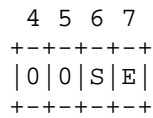


Figure 9: Format of DATA_SEGMENT Header flags

The flags portion of the message header octet contains two optional values in the two low-order bits, denoted 'S' and 'E' in Figure 9. The 'S' bit MUST be set to one if it precedes the transmission of the first segment of a transfer. The 'E' bit MUST be set to one when transmitting the last segment of a transfer. In the case where an entire transfer is accomplished in a single segment, both the 'S' and 'E' bits MUST be set to one.

Following the message header, the length field is a 64-bit unsigned integer containing the number of octets of bundle data that are transmitted in this segment. Following the length are the actual data contents.

Once a transfer of a bundle has commenced, the node MUST only send segments containing sequential portions of that bundle until it sends a segment with the 'E' bit set. No interleaving of multiple transfers from the same endpoint is possible (within a single TCPCL session).

5.4.4. Bundle Acknowledgments (ACK_SEGMENT)

Although the TCP transport provides reliable transfer of data between transport peers, the typical BSD sockets interface provides no means to inform a sending application of when the receiving application has processed some amount of transmitted data. Thus, after transmitting some data, a Bundle Protocol agent needs an additional mechanism to determine whether the receiving agent has successfully received the segment. To this end, the TCPCL protocol provides feedback messaging whereby a receiving node transmits acknowledgments of reception of data segments.

The format of an ACK_SEGMENT message follows in Figure 10 and its use of header flags is the same as for DATA_SEGMENT (shown in Figure 9). The flags of an ACK_SEGMENT message SHALL be identical to the flags of the DATA_SEGMENT message for which it is a reply.

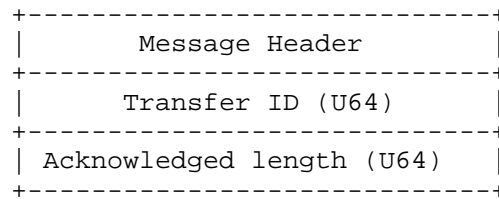


Figure 10: Format of ACK_SEGMENT Messages

A receiving TCPCL endpoint SHALL send an ACK_SEGMENT message in response to each received DATA_SEGMENT message. The flags portion of the ACK_SEGMENT header SHALL be set to match the corresponding DATA_SEGMENT message being acknowledged. The acknowledged length of each ACK_SEGMENT contains the sum of the data length fields of all DATA_SEGMENT messages received so far in the course of the indicated transfer.

For example, suppose the sending node transmits four segments of bundle data with lengths 100, 200, 500, and 1000, respectively. After receiving the first segment, the node sends an acknowledgment of length 100. After the second segment is received, the node sends an acknowledgment of length 300. The third and fourth acknowledgments are of length 800 and 1800, respectively.

5.4.5. Bundle Refusal (REFUSE_BUNDLE)

As bundles can be large, the TCPCL supports an optional mechanism by which a receiving node MAY indicate to the sender that it does not want to receive the corresponding bundle.

To do so, upon receiving a LENGTH or DATA_SEGMENT message, the node MAY transmit a REFUSE_BUNDLE message. As data segments and acknowledgments MAY cross on the wire, the bundle that is being refused SHALL be identified by the Transfer ID of the refusal.

There is no required relation between the Transfer MRU of a TCPCL endpoint (which is supposed to represent a firm limitation of what the endpoint will accept) and sending of a REFUSE_BUNDLE message. A REFUSE_BUNDLE can be used in cases where the agent's bundle storage is temporarily depleted or somehow constrained. A REFUSE_BUNDLE can also be used after the bundle header or any bundle data is inspected by an agent and determined to be unacceptable.

The format of the REFUSE_BUNDLE message is as follows:

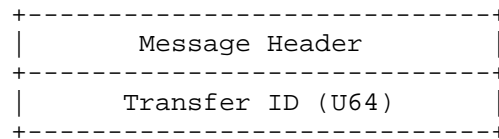


Figure 11: Format of REFUSE_BUNDLE Messages

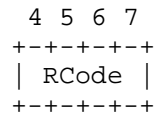


Figure 12: Format of REFUSE_BUNDLE Header flags

The RCode field, which stands for "reason code", contains a value indicating why the bundle was refused. The following table contains semantics for some values. Other values MAY be registered with IANA, as defined in Section 8.

Name	RCode	Semantics
Unknown	0x0	Reason for refusal is unknown or not specified.
Completed	0x1	The receiver now has the complete bundle. The sender MAY now consider the bundle as completely received.
No Resources	0x2	The receiver's resources are exhausted. The sender SHOULD apply reactive bundle fragmentation before retrying.
Retransmit	0x3	The receiver has encountered a problem that requires the bundle to be retransmitted in its entirety.

Table 4: REFUSE_BUNDLE Reason Codes

The receiver MUST, for each transfer preceding the one to be refused, have either acknowledged all DATA_SEGMENTS or refused the bundle transfer.

The bundle transfer refusal MAY be sent before an entire data segment is received. If a sender receives a REFUSE_BUNDLE message, the sender MUST complete the transmission of any partially sent

DATA_SEGMENT message. There is no way to interrupt an individual TCPCL message partway through sending it. The sender MUST NOT commence transmission of any further segments of the refused bundle subsequently. Note, however, that this requirement does not ensure that a node will not receive another DATA_SEGMENT for the same bundle after transmitting a REFUSE_BUNDLE message since messages MAY cross on the wire; if this happens, subsequent segments of the bundle SHOULD also be refused with a REFUSE_BUNDLE message.

Note: If a bundle transmission is aborted in this way, the receiver MAY not receive a segment with the 'E' flag set to '1' for the aborted bundle. The beginning of the next bundle is identified by the 'S' bit set to '1', indicating the start of a new transfer, and with a distinct Transfer ID value.

6. Session Termination

This section describes the procedures for ending a TCPCL session.

6.1. Shutdown Message (SHUTDOWN)

To cleanly shut down a session, a SHUTDOWN message MUST be transmitted by either node at any point following complete transmission of any other message. A receiving node SHOULD acknowledge all received data segments before sending a SHUTDOWN message to end the session. A transmitting node SHALL treat a SHUTDOWN message received mid-transfer (i.e. before the final acknowledgement) as a failure of the transfer.

The format of the SHUTDOWN message is as follows:

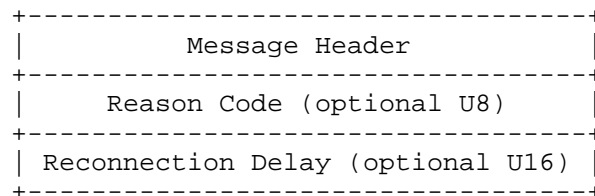


Figure 13: Format of SHUTDOWN Messages

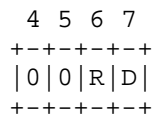


Figure 14: Format of SHUTDOWN Header flags

It is possible for a node to convey additional information regarding the reason for session termination. To do so, the node **MUST** set the 'R' bit in the message header flags and transmit a one-octet reason code immediately following the message header. The specified values of the reason code are:

Name	Code	Description
Idle timeout	0x00	The session is being closed due to idleness.
Version mismatch	0x01	The node cannot conform to the specified TCPCL protocol version.
Busy	0x02	The node is too busy to handle the current session.
Contact Failure	0x03	The node cannot interpret or negotiate contact header option.
TLS failure	0x04	The node failed to negotiate TLS session and cannot continue the session.
Resource Exhaustion	0x05	The node has run into some resource limit and cannot continue the session.

Table 5: SHUTDOWN Reason Codes

It is also possible to convey a requested reconnection delay to indicate how long the other node **MUST** wait before attempting session re-establishment. To do so, the node sets the 'D' bit in the message header flags and then transmits an 16-bit unsigned integer specifying the requested delay, in seconds, following the message header (and optionally, the SHUTDOWN reason code). The value 0 **SHALL** be interpreted as an infinite delay, i.e., that the connecting node **MUST NOT** re-establish the session. In contrast, if the node does not wish to request a delay, it **SHOULD** omit the reconnection delay field (and set the 'D' bit to zero).

A session shutdown **MAY** occur immediately after TCP connection establishment or reception of a contact header (and prior to any further data exchange). This **MAY**, for example, be used to notify that the node is currently not able or willing to communicate. However, a node **MUST** always send the contact header to its peer before sending a SHUTDOWN message.

If either node terminates a session prematurely in this manner, it SHOULD send a SHUTDOWN message and MUST indicate a reason code unless the incoming connection did not include the magic string. If the magic string was not present, a node SHOULD close the TCP connection without sending a SHUTDOWN message. If a node does not want its peer to reopen a connection immediately, it SHOULD set the 'D' bit in the flags and include a reconnection delay to indicate when the peer is allowed to attempt another session setup.

If a session is to be terminated before another protocol message has completed being sent, then the node MUST NOT transmit the SHUTDOWN message but still SHOULD close the TCP connection. This means that a SHUTDOWN cannot be used to preempt any other TCPCL messaging in-progress (particularly important when large segment sizes are being transmitted).

6.2. Idle Session Shutdown

The protocol includes a provision for clean shutdown of idle sessions. Determining the length of time to wait before closing idle sessions, if they are to be closed at all, is an implementation and configuration matter.

If there is a configured time to close idle links and if no bundle data (other than KEEPALIVE messages) has been received for at least that amount of time, then either node MAY terminate the session by transmitting a SHUTDOWN message indicating the reason code of 'Idle timeout' (as described in Table 5). After receiving a SHUTDOWN message in response, both sides MAY close the TCP connection.

7. Security Considerations

One security consideration for this protocol relates to the fact that nodes present their endpoint identifier as part of the contact header exchange. It would be possible for a node to fake this value and present the identity of a singleton endpoint in which the node is not a member, essentially masquerading as another DTN node. If this identifier is used outside of a TLS-secured session or without further verification as a means to determine which bundles are transmitted over the session, then the node that has falsified its identity would be able to obtain bundles that it otherwise would not have. Therefore, a node SHALL NOT use the EID value of an unsecured contact header to derive a peer node's identity unless it can corroborate it via other means. When TCPCL session security is mandatory, an endpoint SHALL transmit initial unsecured contact header values indicated in Table 6 in order. These values avoid unnecessarily leaking endpoint parameters and will be ignored when secure contact header re-exchange occurs.

Parameter	Value
Flags	The USE_TLS flag is set.
Keepalive Interval	Zero, indicating no keepalive.
Segment MRU	Zero, indicating all segments are refused.
Transfer MRU	Zero, indicating all transfers are refused.
EID	Empty, indicating lack of EID.

Table 6: Recommended Unsecured Contact Header

TCPCL can be used to provide point-to-point transport security, but does not provide security of data-at-rest and does not guarantee end-to-end bundle security. The mechanisms defined in [RFC6257] and [I-D.ietf-dtn-bpsec] are to be used instead.

Even when using TLS to secure the TCPCL session, the actual ciphersuite negotiated between the TLS peers MAY be insecure. TLS can be used to perform authentication without data confidentiality, for example. It is up to security policies within each TCPCL node to ensure that the negotiated TLS ciphersuite meets transport security requirements. This is identical behavior to STARTTLS use in [RFC2595].

Another consideration for this protocol relates to denial-of-service attacks. A node MAY send a large amount of data over a TCPCL session, requiring the receiving node to handle the data, attempt to stop the flood of data by sending a REFUSE_BUNDLE message, or forcibly terminate the session. This burden could cause denial of service on other, well-behaving sessions. There is also nothing to prevent a malicious node from continually establishing sessions and repeatedly trying to send copious amounts of bundle data. A listening node MAY take countermeasures such as ignoring TCP SYN messages, closing TCP connections as soon as they are established, waiting before sending the contact header, sending a SHUTDOWN message quickly or with a delay, etc.

8. IANA Considerations

In this section, registration procedures are as defined in [RFC5226]

8.1. Port Number

Port number 4556 has been previously assigned as the default port for the TCP convergence layer in [RFC7242]. This assignment is unchanged by protocol version 4.

Parameter	Value
Service Name:	dtn-bundle
Transport Protocol(s):	TCP
Assignee:	Simon Perreault <simon@per.reau.lt>
Contact:	Simon Perreault <simon@per.reau.lt>
Description:	DTN Bundle TCP CL Protocol
Reference:	[RFC7242]
Port Number:	4556

8.2. Protocol Versions

IANA has created, under the "Bundle Protocol" registry, a sub-registry titled "Bundle Protocol TCP Convergence-Layer Version Numbers" and initialized it with the following table. The registration procedure is RFC Required.

Value	Description	Reference
0	Reserved	[RFC7242]
1	Reserved	[RFC7242]
2	Reserved	[RFC7242]
3	TCPCL	[RFC7242]
4	TCPCLbis	This specification.
5-255	Unassigned	

8.3. Message Types

IANA has created, under the "Bundle Protocol" registry, a sub-registry titled "Bundle Protocol TCP Convergence-Layer Message Types" and initialized it with the contents below. The registration procedure is RFC Required.

Code	Message Type
0x0	Reserved
0x1	DATA_SEGMENT
0x2	ACK_SEGMENT
0x3	REFUSE_BUNDLE
0x4	KEEPALIVE
0x5	SHUTDOWN
0x6	LENGTH
TBD	REJECT
TBD--0xf	Unassigned

Message Type Codes

8.4. REFUSE_BUNDLE Reason Codes

IANA has created, under the "Bundle Protocol" registry, a sub-registry titled "Bundle Protocol TCP Convergence-Layer REFUSE_BUNDLE Reason Codes" and initialized it with the contents of Table 3. The registration procedure is RFC Required.

Code	Refusal Reason
0x0	Unknown
0x1	Completed
0x2	No Resources
0x3	Retransmit
0x4--0x7	Unassigned
0x8--0xf	Reserved for future usage

REFUSE_BUNDLE Reason Codes

8.5. SHUTDOWN Reason Codes

IANA has created, under the "Bundle Protocol" registry, a sub-registry titled "Bundle Protocol TCP Convergence-Layer SHUTDOWN Reason Codes" and initialized it with the contents of Table 4. The registration procedure is RFC Required.

Code	Shutdown Reason
0x00	Idle timeout
0x01	Version mismatch
0x02	Busy
TBD	Contact Failure
TBD	TLS failure
TBD--0xFF	Unassigned

SHUTDOWN Reason Codes

8.6. REJECT Reason Codes

EDITOR NOTE: sub-registry to-be-created upon publication of this specification.

IANA will create, under the "Bundle Protocol" registry, a sub-registry titled "Bundle Protocol TCP Convergence-Layer REJECT Reason Codes" and initialized it with the contents of Table 4. The registration procedure is RFC Required.

Code	Rejection Reason
0x00	reserved
0x01	Message Type Unknown
0x02	Message Unsupported
0x03	Message Unexpected
0x04-0xFF	Unassigned

REJECT Reason Codes

9. Acknowledgments

This memo is based on comments on implementation of [RFC7242] provided from Scott Burleigh.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC5050] Scott, K. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, DOI 10.17487/RFC5050, November 2007, <<http://www.rfc-editor.org/info/rfc5050>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.

- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<http://www.rfc-editor.org/info/rfc7525>>.
- [I-D.ietf-dtn-bpbis] Burleigh, S., Fall, K., and E. Birrane, "Bundle Protocol", draft-ietf-dtn-bpbis-06 (work in progress), October 2016.
- [refs.IANA-BP] IANA, "Bundle Protocol registry", May 2016.

10.2. Informative References

- [RFC2595] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, DOI 10.17487/RFC2595, June 1999, <<http://www.rfc-editor.org/info/rfc2595>>.
- [RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <<http://www.rfc-editor.org/info/rfc4838>>.
- [RFC6257] Symington, S., Farrell, S., Weiss, H., and P. Lovell, "Bundle Security Protocol Specification", RFC 6257, DOI 10.17487/RFC6257, May 2011, <<http://www.rfc-editor.org/info/rfc6257>>.
- [RFC7242] Demmer, M., Ott, J., and S. Perreault, "Delay-Tolerant Networking TCP Convergence-Layer Protocol", RFC 7242, DOI 10.17487/RFC7242, June 2014, <<http://www.rfc-editor.org/info/rfc7242>>.
- [I-D.ietf-dtn-bpsec] Birrane, E. and K. McKeever, "Bundle Protocol Security Specification", draft-ietf-dtn-bpsec-03 (work in progress), October 2016.

Appendix A. Significant changes from RFC7242

The areas in which changes from [RFC7242] have been made to existing messages are:

- o Changed contact header content to limit number of negotiated options.

- o Added contact option to negotiate maximum segment size (per each direction).
- o Added a bundle transfer identification number to all bundle-related messages (LENGTH, DATA_SEGMENT, ACK_SEGMENT, REFUSE_BUNDLE).
- o Use flags in ACK_SEGMENT to mirror flags from DATA_SEGMENT.
- o Removed all uses of SDNV fields and replaced with fixed-bit-length fields.

The areas in which extensions from [RFC7242] have been made as new messages and codes are:

- o Added contact negotiation failure SHUTDOWN reason code.
- o Added REJECT message to indicate an unknown or unhandled message was received.
- o Added TLS session security mechanism.
- o Added TLS failure SHUTDOWN reason code.

Authors' Addresses

Brian Sipos
RKF Engineering Solutions, LLC
1229 19th Street NW
Washington, DC 20036
US

Email: BSipos@rkf-eng.com

Michael Demmer
University of California, Berkeley
Computer Science Division
445 Soda Hall
Berkeley, CA 94720-1776
US

Email: demmer@cs.berkeley.edu

Joerg Ott
Aalto University
Department of Communications and Networking
PO Box 13000
Aalto 02015
Finland

Email: jo@netlab.tkk.fi

Simon Perreault
Quebec, QC
Canada

Email: simon@per.reau.lt

Delay-Tolerant Networking
Internet-Draft
Intended status: Standards Track
Expires: 9 April 2022

B. Sipos
RKF Engineering
M. Demmer
UC Berkeley
J. Ott
Aalto University
S. Perreault
6 October 2021

Delay-Tolerant Networking TCP Convergence Layer Protocol Version 4
draft-ietf-dtn-tcpclv4-28

Abstract

This document describes a TCP-based convergence layer (TCPCL) for Delay-Tolerant Networking (DTN). This version of the TCPCL protocol resolves implementation issues in the earlier TCPCL Version 3 of RFC7242 and updates to the Bundle Protocol (BP) contents, encodings, and convergence layer requirements in BP Version 7. Specifically, the TCPCLv4 uses CBOR-encoded BPv7 bundles as its service data unit being transported and provides a reliable transport of such bundles. This version of TCPCL also includes security and extensibility mechanisms.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 9 April 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Scope	5
2. Requirements Language	5
2.1. Definitions Specific to the TCPCL Protocol	6
3. General Protocol Description	9
3.1. Convergence Layer Services	9
3.2. TCPCL Session Overview	12
3.3. TCPCL States and Transitions	13
3.4. PKIX Environments and CA Policy	19
3.5. Session Keeping Policies	20
3.6. Transfer Segmentation Policies	21
3.7. Example Message Exchange	22
4. Session Establishment	24
4.1. TCP Connection	24
4.2. Contact Header	25
4.3. Contact Validation and Negotiation	26
4.4. Session Security	28
4.4.1. Entity Identification	28
4.4.2. Certificate Profile for TCPCL	29
4.4.3. TLS Handshake	31
4.4.4. TLS Authentication	32
4.4.5. Policy Recommendations	34
4.4.6. Example TLS Initiation	34
4.5. Message Header	36
4.6. Session Initialization Message (SESS_INIT)	37
4.7. Session Parameter Negotiation	39
4.8. Session Extension Items	40
5. Established Session Operation	41
5.1. Upkeep and Status Messages	41
5.1.1. Session Upkeep (KEEPALIVE)	41
5.1.2. Message Rejection (MSG_REJECT)	42
5.2. Bundle Transfer	44
5.2.1. Bundle Transfer ID	45
5.2.2. Data Transmission (XFER_SEGMENT)	45
5.2.3. Data Acknowledgments (XFER_ACK)	47
5.2.4. Transfer Refusal (XFER_REFUSE)	49
5.2.5. Transfer Extension Items	51

6.	Session Termination	53
6.1.	Session Termination Message (SESS_TERM)	53
6.2.	Idle Session Shutdown	56
7.	Implementation Status	56
8.	Security Considerations	57
8.1.	Threat: Passive Leak of Node Data	57
8.2.	Threat: Passive Leak of Bundle Data	57
8.3.	Threat: TCPCL Version Downgrade	57
8.4.	Threat: Transport Security Stripping	57
8.5.	Threat: Weak TLS Configurations	58
8.6.	Threat: Untrusted End-Entity Certificate	58
8.7.	Threat: Certificate Validation Vulnerabilities	58
8.8.	Threat: Symmetric Key Limits	59
8.9.	Threat: BP Node Impersonation	59
8.10.	Threat: Denial of Service	60
8.11.	Mandatory-to-Implement TLS	61
8.12.	Alternate Uses of TLS	61
8.12.1.	TLS Without Authentication	61
8.12.2.	Non-Certificate TLS Use	61
8.13.	Predictability of Transfer IDs	62
9.	IANA Considerations	62
9.1.	Port Number	62
9.2.	Protocol Versions	63
9.3.	Session Extension Types	64
9.4.	Transfer Extension Types	64
9.5.	Message Types	65
9.6.	XFER_REFUSE Reason Codes	66
9.7.	SESS_TERM Reason Codes	67
9.8.	MSG_REJECT Reason Codes	68
9.9.	Object Identifier for PKIX Module Identifier	69
9.10.	Object Identifier for PKIX Other Name Forms	69
9.11.	Object Identifier for PKIX Extended Key Usage	70
10.	Acknowledgments	70
11.	References	70
11.1.	Normative References	70
11.2.	Informative References	72
	Appendix A. Significant changes from RFC7242	74
	Appendix B. ASN.1 Module	76
	Appendix C. Example of the BundleEID Other Name Form	78
	Authors' Addresses	78

1. Introduction

This document describes the TCP-based convergence-layer protocol for Delay-Tolerant Networking. Delay-Tolerant Networking is an end-to-end architecture providing communications in and/or through highly stressed environments, including those with intermittent connectivity, long and/or variable delays, and high bit error rates. More detailed descriptions of the rationale and capabilities of these networks can be found in "Delay-Tolerant Network Architecture" [RFC4838].

An important goal of the DTN architecture is to accommodate a wide range of networking technologies and environments. The protocol used for DTN communications is the Bundle Protocol Version 7 (BPv7) [I-D.ietf-dtn-bpbis], an application-layer protocol that is used to construct a store-and-forward overlay network. BPv7 requires the services of a "convergence-layer adapter" (CLA) to send and receive bundles using the service of some "native" link, network, or Internet protocol. This document describes one such convergence-layer adapter that uses the well-known Transmission Control Protocol (TCP). This convergence layer is referred to as TCP Convergence Layer Version 4 (TCPCLv4). For the remainder of this document, the abbreviation "BP" without the version suffix refers to BPv7. For the remainder of this document, the abbreviation "TCPCL" without the version suffix refers to TCPCLv4.

The locations of the TCPCL and the BP in the Internet model protocol stack (described in [RFC1122]) are shown in Figure 1. In particular, when BP is using TCP as its bearer with TCPCL as its convergence layer, both BP and TCPCL reside at the application layer of the Internet model.

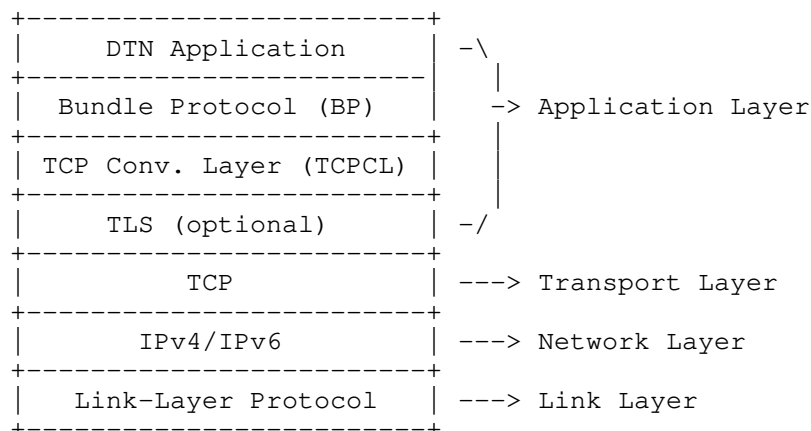


Figure 1: The Locations of the Bundle Protocol and the TCP Convergence-Layer Protocol above the Internet Protocol Stack

1.1. Scope

This document describes the format of the protocol data units passed between entities participating in TCPCL communications. This document does not address:

- * The format of protocol data units of the Bundle Protocol, as those are defined elsewhere in [I-D.ietf-dtn-bpbis]. This includes the concept of bundle fragmentation or bundle encapsulation. The TCPCL transfers bundles as opaque data blocks.
- * Mechanisms for locating or identifying other bundle entities (peers) within a network or across an internet. The mapping of Node ID to potential convergence layer (CL) protocol and network address is left to implementation and configuration of the BP Agent and its various potential routing strategies. The mapping of DNS name and/or address to a choice of end-entity certificate to authenticate a node to its peers.
- * Logic for routing bundles along a path toward a bundle's endpoint. This CL protocol is involved only in transporting bundles between adjacent entities in a routing sequence.
- * Policies or mechanisms for issuing Public Key Infrastructure Using X.509 (PKIX) certificates; provisioning, deploying, or accessing certificates and private keys; deploying or accessing certificate revocation lists (CRLs); or configuring security parameters on an individual entity or across a network.
- * Uses of TLS which are not based on PKIX certificate authentication (see Section 8.12.2) or in which authentication of both entities is not possible (see Section 8.12.1).

Any TCPCL implementation requires a BP agent to perform those above listed functions in order to perform end-to-end bundle delivery.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.1. Definitions Specific to the TCPCL Protocol

This section contains definitions specific to the TCPCL protocol.

Network Byte Order: Most significant byte first, a.k.a., big endian. All of the integer encodings in this protocol SHALL be transmitted in network byte order.

TCPCL Entity: This is the notional TCPCL application that initiates TCPCL sessions. This design, implementation, configuration, and specific behavior of such an entity is outside of the scope of this document. However, the concept of an entity has utility within the scope of this document as the container and initiator of TCPCL sessions. The relationship between a TCPCL entity and TCPCL sessions is defined as follows:

- * A TCPCL Entity MAY actively initiate any number of TCPCL Sessions and should do so whenever the entity is the initial transmitter of information to another entity in the network.
- * A TCPCL Entity MAY support zero or more passive listening elements that listen for connection requests from other TCPCL Entities operating on other entities in the network.
- * A TCPCL Entity MAY passively initiate any number of TCPCL Sessions from requests received by its passive listening element(s) if the entity uses such elements.

These relationships are illustrated in Figure 2. For most TCPCL behavior within a session, the two entities are symmetric and there is no protocol distinction between them. Some specific behavior, particularly during session establishment, distinguishes between the active entity and the passive entity. For the remainder of this document, the term "entity" without the prefix "TCPCL" refers to a TCPCL entity.

TCP Connection: The term Connection in this specification exclusively refers to a TCP connection and any and all behaviors, sessions, and other states associated with that TCP connection.

TCPCL Session: A TCPCL session (as opposed to a TCP connection) is a TCPCL communication relationship between two TCPCL entities. A TCPCL session operates within a single underlying TCP connection and the lifetime of a TCPCL session is bound to the lifetime of that TCP connection. A TCPCL session is terminated when the TCP connection ends, due either to one or both entities actively closing the TCP connection or due to network errors causing a failure of the TCP connection. Within a single TCPCL session

there are two possible transfer streams; one in each direction, with one stream from each entity being the outbound stream and the other being the inbound stream (see Figure 3). From the perspective of a TCPCL session, the two transfer streams do not logically interact with each other. The streams do operate over the same TCP connection and between the same BP agents, so there are logical relationships at those layers (message and bundle interleaving respectively). For the remainder of this document, the term "session" without the prefix "TCPCL" refers to a TCPCL session.

Session parameters: These are a set of values used to affect the operation of the TCPCL for a given session. The manner in which these parameters are conveyed to the bundle entity and thereby to the TCPCL is implementation dependent. However, the mechanism by which two entities exchange and negotiate the values to be used for a given session is described in Section 4.3.

Transfer Stream: A Transfer stream is a uni-directional user-data path within a TCPCL Session. Transfers sent over a transfer stream are serialized, meaning that one transfer must complete its transmission prior to another transfer being started over the same transfer stream. At the stream layer there is no logical relationship between transfers in that stream; it's only within the BP agent that transfers are fully decoded as bundles. Each uni-directional stream has a single sender entity and a single receiver entity.

Transfer: This refers to the procedures and mechanisms for conveyance of an individual bundle from one node to another. Each transfer within TCPCL is identified by a Transfer ID number which is guaranteed to be unique only to a single direction within a single Session.

Transfer Segment: A subset of a transfer of user data being communicated over a transfer stream.

Idle Session: A TCPCL session is idle while there is no transmission in-progress in either direction. While idle, the only messages being transmitted or received are KEEPALIVE messages.

Live Session: A TCPCL session is live while there is a transmission in-progress in either direction.

Reason Codes: The TCPCL uses numeric codes to encode specific reasons for individual failure/error message types.

The relationship between connections, sessions, and streams is shown in Figure 3.

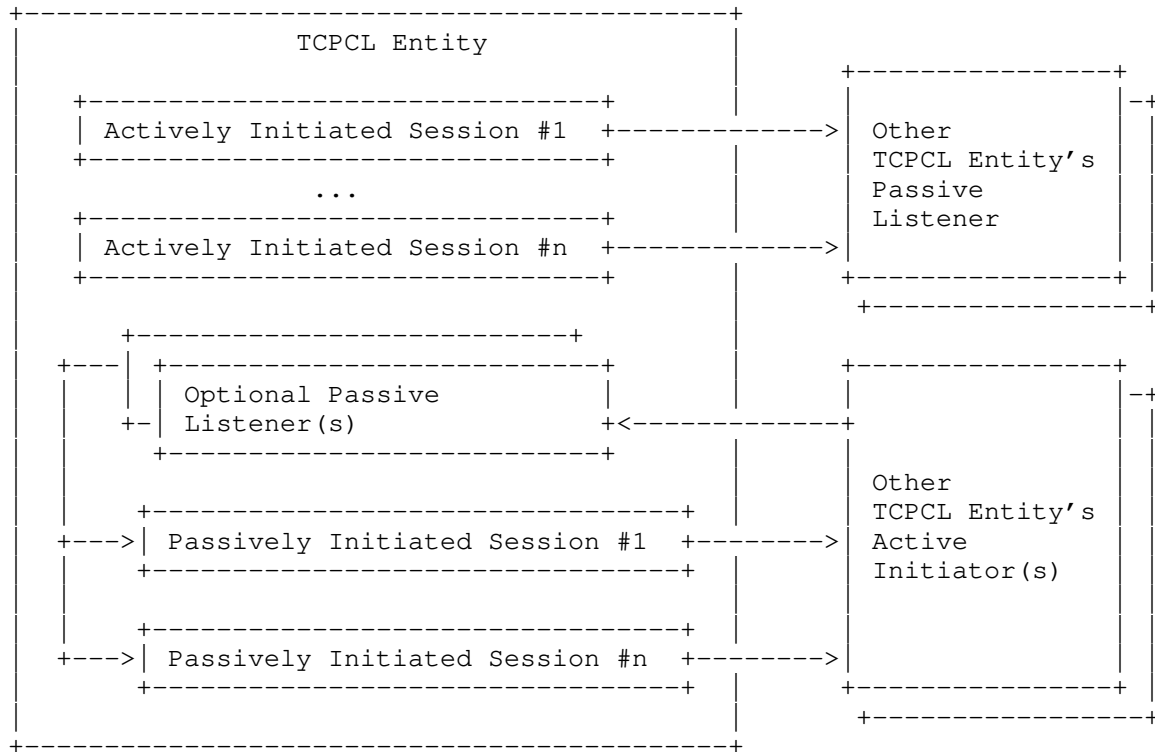


Figure 2: The relationships between TCPCL entities

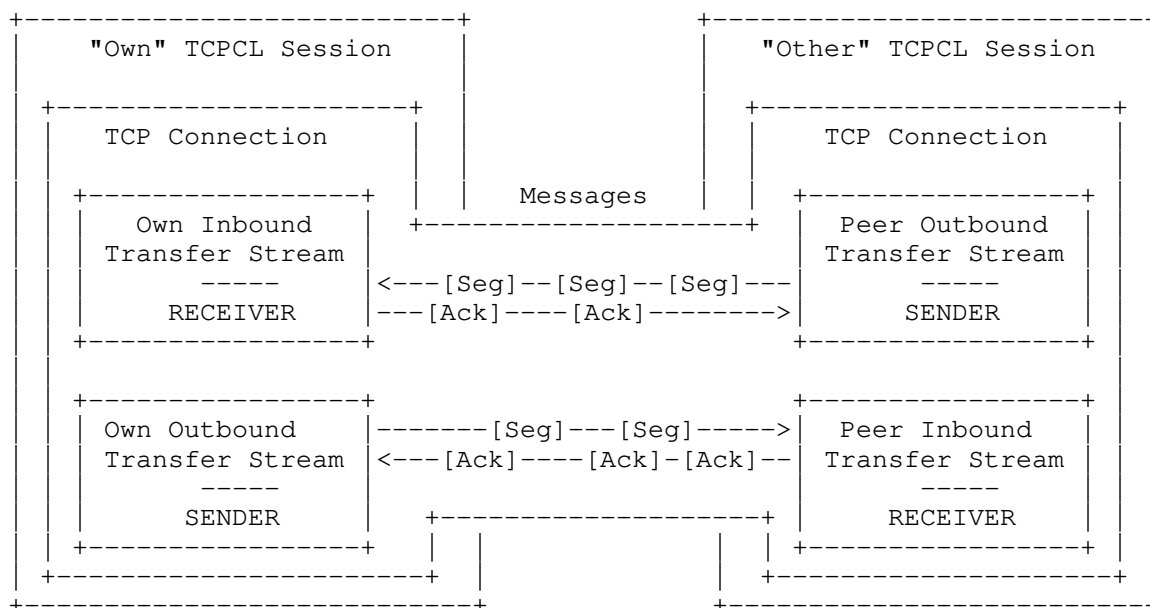


Figure 3: The relationship within a TCPCL Session of its two streams

3. General Protocol Description

The service of this protocol is the transmission of DTN bundles via the Transmission Control Protocol (TCP). This document specifies the encapsulation of bundles, procedures for TCP setup and teardown, and a set of messages and entity requirements. The general operation of the protocol is as follows.

3.1. Convergence Layer Services

This version of the TCPCL provides the following services to support the overlaying Bundle Protocol agent. In all cases, this is not an API definition but a logical description of how the CL can interact with the BP agent. Each of these interactions can be associated with any number of additional metadata items as necessary to support the operation of the CL or BP agent.

Attempt Session: The TCPCL allows a BP agent to preemptively attempt to establish a TCPCL session with a peer entity. Each session attempt can send a different set of session negotiation parameters as directed by the BP agent.

Terminate Session: The TCPCL allows a BP agent to preemptively

terminate an established TCPCL session with a peer entity. The terminate request is on a per-session basis.

Session State Changed: The TCPCL entity indicates to the BP agent when the session state changes. The top-level session states indicated are:

Connecting: A TCP connection is being established. This state only applies to the active entity.

Contact Negotiating: A TCP connection has been made (as either active or passive entity) and contact negotiation has begun.

Session Negotiating: Contact negotiation has been completed (including possible TLS use) and session negotiation has begun.

Established: The session has been fully established and is ready for its first transfer. When the session is established, the peer Node ID (along with indication of whether or not it was authenticated) and the negotiated session parameters (see Section 4.7) are also communicated to the BP agent.

Ending: The entity sent SESS_TERM message and is in the ending state.

Terminated: The session has finished normal termination sequencing.

Failed: The session ended without normal termination sequencing.

Session Idle Changed: The TCPCL entity indicates to the BP agent when the live/idle sub-state of the session changes. This occurs only when the top-level session state is "Established". The session transitions from Idle to Live at the at the start of a transfer in either transfer stream; the session transitions from Live to Idle at the end of a transfer when the other transfer stream does not have an ongoing transfer. Because TCPCL transmits serially over a TCP connection it suffers from "head of queue blocking," so a transfer in either direction can block an immediate start of a new transfer in the session.

Begin Transmission: The principal purpose of the TCPCL is to allow a BP agent to transmit bundle data over an established TCPCL session. Transmission request is on a per-session basis and the CL does not necessarily perform any per-session or inter-session queueing. Any queueing of transmissions is the obligation of the BP agent.

Transmission Success: The TCPCL entity indicates to the BP agent when a bundle has been fully transferred to a peer entity.

Transmission Intermediate Progress: The TCPCL entity indicates to the BP agent on intermediate progress of transfer to a peer entity. This intermediate progress is at the granularity of each transferred segment.

Transmission Failure: The TCPCL entity indicates to the BP agent on certain reasons for bundle transmission failure, notably when the peer entity rejects the bundle or when a TCPCL session ends before transfer success. The TCPCL itself does not have a notion of transfer timeout.

Reception Initialized: The TCPCL entity indicates to the receiving BP agent just before any transmission data is sent. This corresponds to reception of the XFER_SEGMENT message with the START flag of 1.

Interrupt Reception: The TCPCL entity allows a BP agent to interrupt an individual transfer before it has fully completed (successfully or not). Interruption can occur any time after the reception is initialized.

Reception Success: The TCPCL entity indicates to the BP agent when a bundle has been fully transferred from a peer entity.

Reception Intermediate Progress: The TCPCL entity indicates to the BP agent on intermediate progress of transfer from the peer entity. This intermediate progress is at the granularity of each transferred segment. Intermediate reception indication allows a BP agent the chance to inspect bundle header contents before the entire bundle is available, and thus supports the "Reception Interruption" capability.

Reception Failure: The TCPCL entity indicates to the BP agent on certain reasons for reception failure, notably when the local entity rejects an attempted transfer for some local policy reason or when a TCPCL session ends before transfer success. The TCPCL itself does not have a notion of transfer timeout.

3.2. TCPCL Session Overview

First, one entity establishes a TCPCL session to the other by initiating a TCP connection in accordance with [RFC0793]. After setup of the TCP connection is complete, an initial Contact Header is exchanged in both directions to establish a shared TCPCL version and negotiate the use of TLS security (as described in Section 4). Once contact negotiation is complete, TCPCL messaging is available and the session negotiation is used to set parameters of the TCPCL session. One of these parameters is a Node ID that each TCPCL Entity is acting as. This is used to assist in routing and forwarding messages by the BP Agent and is part of the authentication capability provided by TLS.

Once negotiated, the parameters of a TCPCL session cannot change and if there is a desire by either peer to transfer data under different parameters then a new session must be established. This makes CL logic simpler but relies on the assumption that establishing a TCP connection is lightweight enough that TCP connection overhead is negligible compared to TCPCL data sizes.

Once the TCPCL session is established and configured in this way, bundles can be transferred in either direction. Each transfer is performed by segmenting the transfer data into one or more XFER_SEGMENT messages. Multiple bundles can be transmitted consecutively in a single direction on a single TCPCL connection. Segments from different bundles are never interleaved. Bundle interleaving can be accomplished by fragmentation at the BP layer or by establishing multiple TCPCL sessions between the same peers. There is no fundamental limit on the number of TCPCL sessions which a single entity can establish beyond the limit imposed by the number of available (ephemeral) TCP ports of the active entity.

A feature of this protocol is for the receiving entity to send acknowledgment (XFER_ACK) messages as bundle data segments arrive. The rationale behind these acknowledgments is to enable the transmitting entity to determine how much of the bundle has been received, so that in case the session is interrupted, it can perform reactive fragmentation to avoid re-sending the already transmitted part of the bundle. In addition, there is no explicit flow control on the TCPCL layer.

A TCPCL receiver can interrupt the transmission of a bundle at any point in time by replying with a XFER_REFUSE message, which causes the sender to stop transmission of the associated bundle (if it hasn't already finished transmission). Note: This enables a cross-layer optimization in that it allows a receiver that detects that it already has received a certain bundle to interrupt transmission as early as possible and thus save transmission capacity for other bundles.

For sessions that are idle, a KEEPALIVE message is sent at a negotiated interval. This is used to convey entity live-ness information during otherwise message-less time intervals.

A SESS_TERM message is used to initiate the ending of a TCPCL session (see Section 6.1). During termination sequencing, in-progress transfers can be completed but no new transfers can be initiated. A SESS_TERM message can also be used to refuse a session setup by a peer (see Section 4.3). Regardless of the reason, session termination is initiated by one of the entities and responded-to by the other as illustrated by Figure 13 and Figure 14. Even when there are no transfers queued or in-progress, the session termination procedure allows each entity to distinguish between a clean end to a session and the TCP connection being closed because of some underlying network issue.

Once a session is established, TCPCL is a symmetric protocol between the peers. Both sides can start sending data segments in a session, and one side's bundle transfer does not have to complete before the other side can start sending data segments on its own. Hence, the protocol allows for a bi-directional mode of communication. Note that in the case of concurrent bidirectional transmission, acknowledgment segments MAY be interleaved with data segments.

3.3. TCPCL States and Transitions

The states of a normal TCPCL session (i.e., without session failures) are indicated in Figure 4.

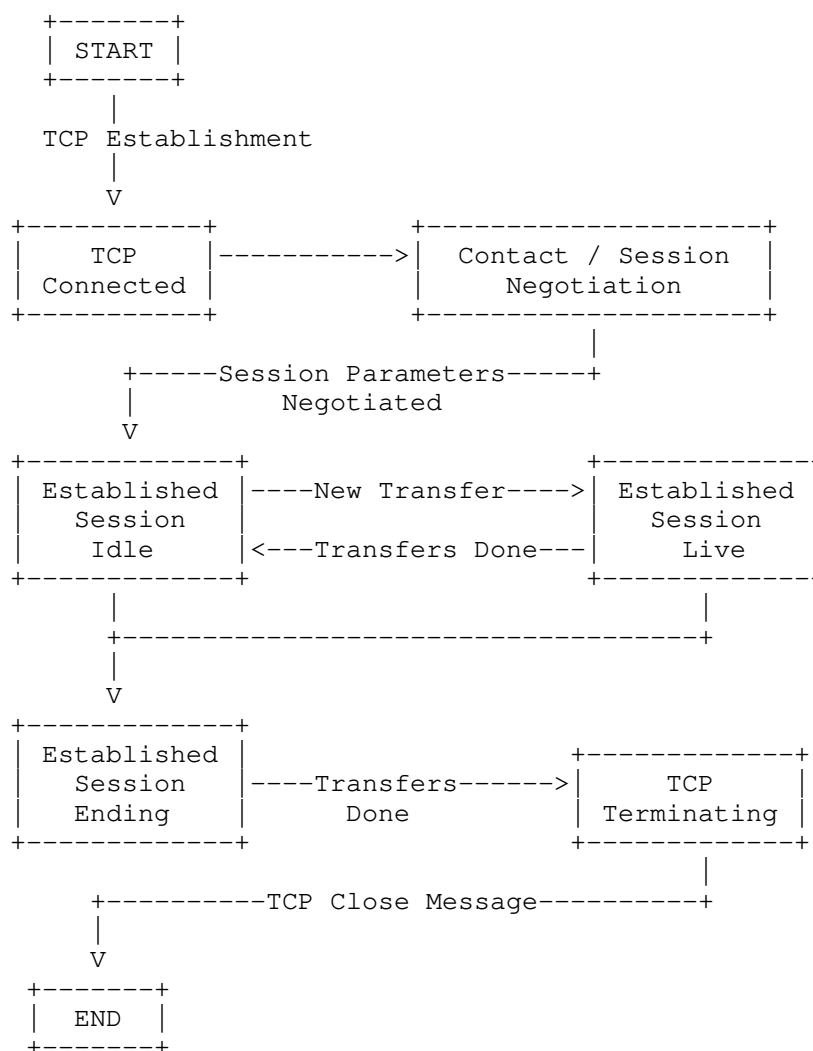


Figure 4: Top-level states of a TCPCL session

Notes on Established Session states:

Session "Live" means transmitting or receiving over a transfer stream.

Session "Idle" means no transmission/reception over a transfer stream.

Session "Ending" means no new transfers will be allowed.

Contact negotiation involves exchanging a Contact Header (CH) in both directions and deriving a negotiated state from the two headers. The contact negotiation sequencing is performed either as the active or passive entity, and is illustrated in Figure 5 and Figure 6 respectively which both share the data validation and negotiation of the Processing of Contact Header "[PCH]" activity of Figure 7 and the "[TCPCLOSE]" activity which indicates TCP connection close. Successful negotiation results in one of the Session Initiation "[SI]" activities being performed. To avoid data loss, a Session Termination "[ST]" exchange allows cleanly finishing transfers before a session is ended.

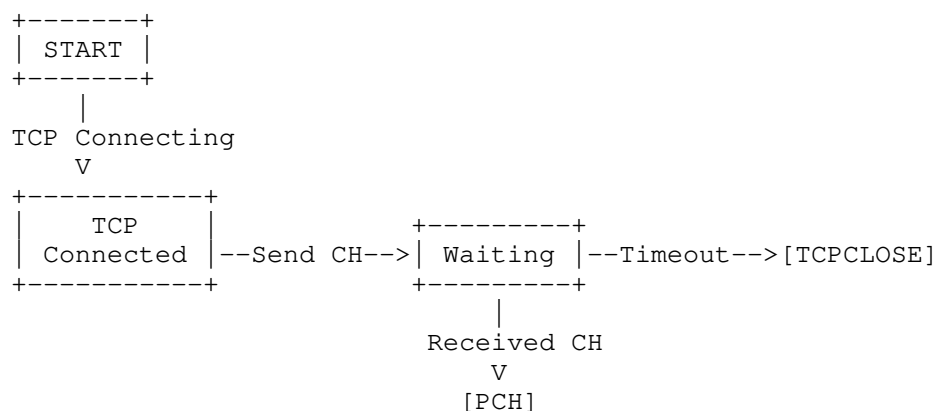


Figure 5: Contact Initiation as Active Entity

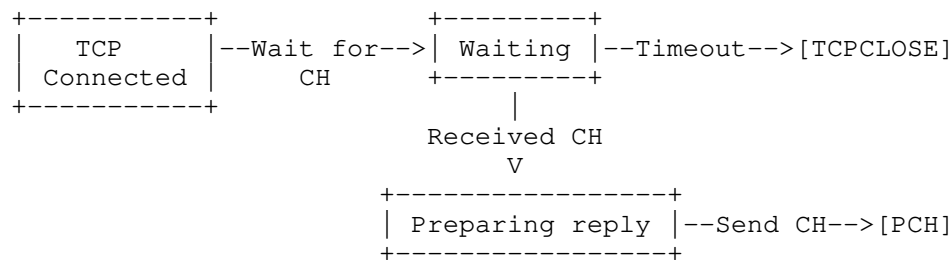


Figure 6: Contact Initiation as Passive Entity

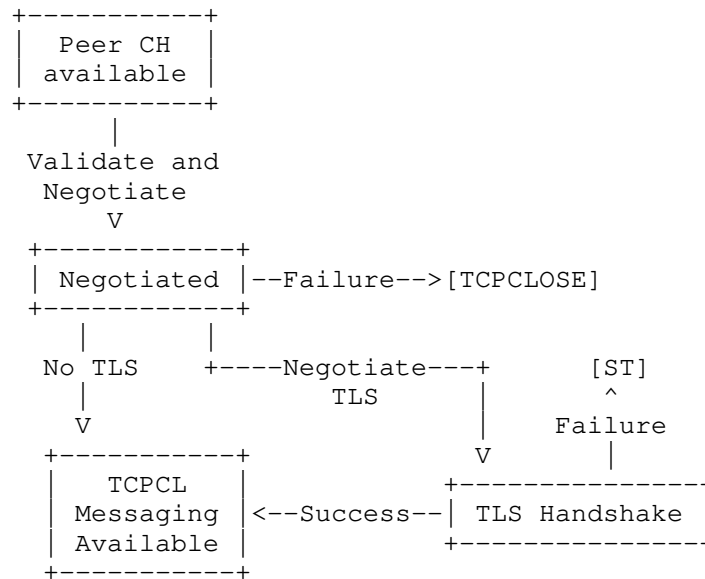


Figure 7: Processing of Contact Header [PCH]

Session negotiation involves exchanging a session initialization (SESS_INIT) message in both directions and deriving a negotiated state from the two messages. The session negotiation sequencing is performed either as the active or passive entity, and is illustrated in Figure 8 and Figure 9 respectively which both share the data validation and negotiation of Figure 10. The validation here includes certificate validation and authentication when TLS is used for the session.

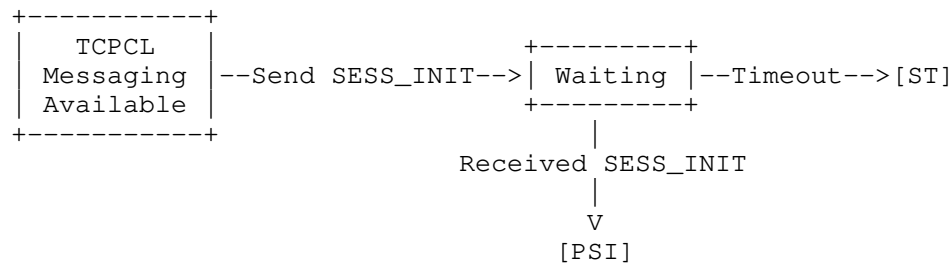


Figure 8: Session Initiation [SI] as Active Entity

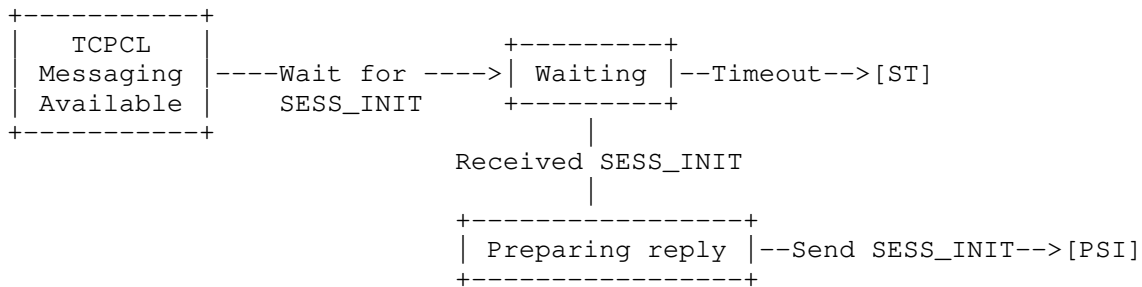


Figure 9: Session Initiation [SI] as Passive Entity

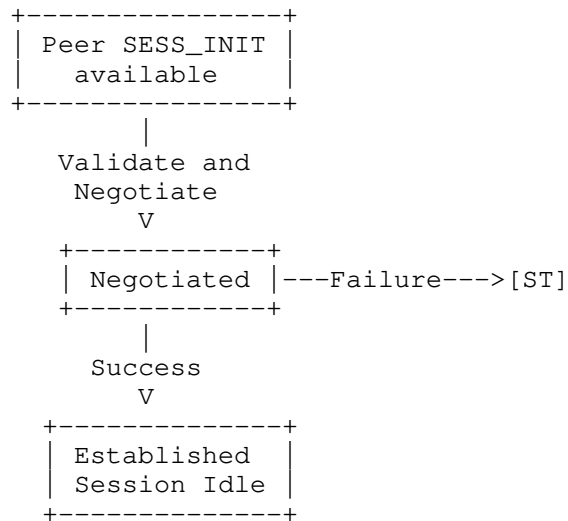


Figure 10: Processing of Session Initiation [PSI]

Transfers can occur after a session is established and it's not in the Ending state. Each transfer occurs within a single logical transfer stream between a sender and a receiver, as illustrated in Figure 11 and Figure 12 respectively.

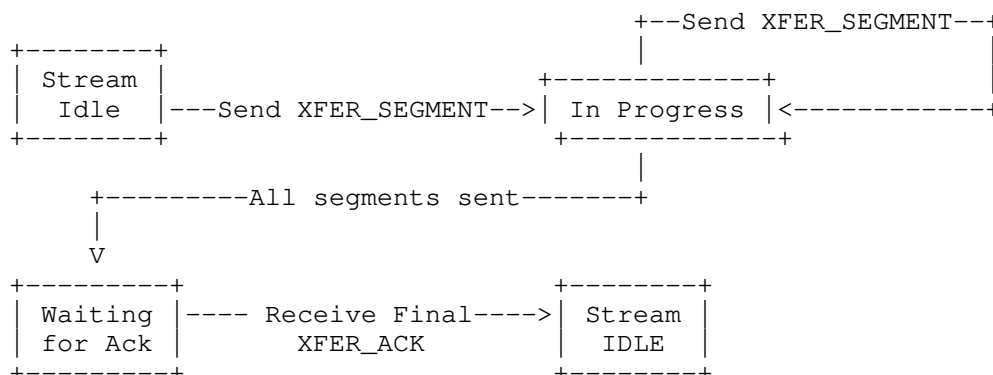


Figure 11: Transfer sender states

Notes on transfer sending:

Pipelining of transfers can occur when the sending entity begins a new transfer while in the "Waiting for Ack" state.

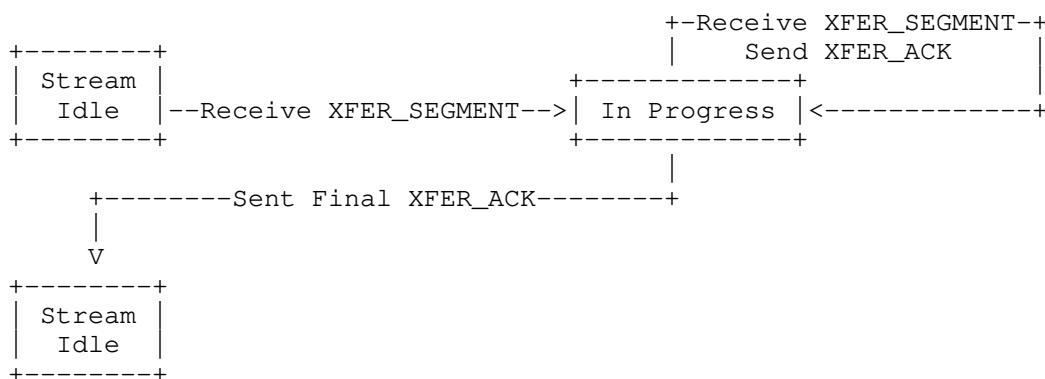


Figure 12: Transfer receiver states

Session termination involves one entity initiating the termination of the session and the other entity acknowledging the termination. For either entity, it is the sending of the SESS_TERM message which transitions the session to the Ending substate. While a session is in the Ending state only in-progress transfers can be completed and no new transfers can be started.

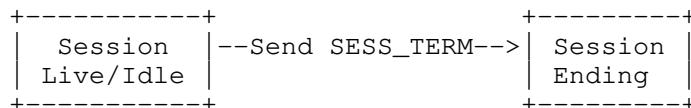


Figure 13: Session Termination [ST] from the Initiator

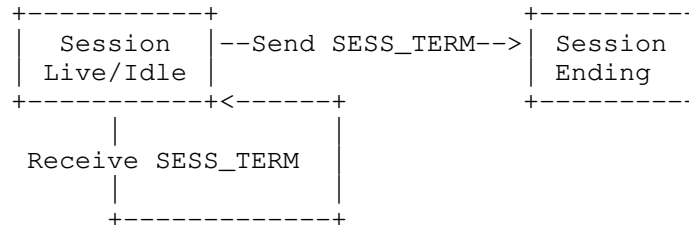


Figure 14: Session Termination [ST] from the Responder

3.4. PKIX Environments and CA Policy

This specification gives requirements about how to use PKIX certificates issued by a Certificate Authority (CA), but does not define any mechanisms for how those certificates come to be. The requirements about TCPCL certificate use are broad to support two quite different PKIX environments:

DTN-Aware CAs: In the ideal case, the CA(s) issuing certificates for TCPCL entities are aware of the end use of the certificate, have a mechanism for verifying ownership of a Node ID, and are issuing certificates directly for that Node ID. In this environment, the ability to authenticate a peer entity Node ID directly avoids the need to authenticate a network name or address and then implicitly trust Node ID of the peer. The TCPCL authenticates the Node ID whenever possible and this is preferred over lower-level PKIX identities.

DTN-Ignorant CAs: It is expected that Internet-scale "public" CAs will continue to focus on DNS names as the preferred PKIX identifier. There are large infrastructures already in-place for managing network-level authentication and protocols to manage identity verification in those environments [RFC8555]. The TCPCL allows for this type of environment by authenticating a lower-level identifier for a peer and requiring the entity to trust that the Node ID given by the peer (during session initialization) is valid. This situation is not ideal, as it allows vulnerabilities described in Section 8.9, but still provides some amount of mutual authentication to take place for a TCPCL session.

Even within a single TCPCL session, each entity may operate within different PKI environments and with different identifier limitations. The requirements related to identifiers in a PKIX certificate are in Section 4.4.1.

It is important for interoperability that a TCPCL entity have its own security policy tailored to accommodate the peers with which it is expected to operate. Some security policy recommendations are given in Section 4.4.5 but these are meant as a starting point for tailoring. A strict TLS security policy is appropriate for a private network with a single shared CA. Operation on the Internet (such as inter-site BP gateways) could trade more lax TCPCL security with the use of encrypted bundle encapsulation [I-D.ietf-dtn-bibect] to ensure strong bundle security.

By using the Server Name Indication (SNI) DNS name (see Section 4.4.3) a single passive entity can act as a convergence layer for multiple BP agents with distinct Node IDs. When this "virtual host" behavior is used, the DNS name is used as the indication of which BP Node the active entity is attempting to communicate with. A virtual host CL entity can be authenticated by a certificate containing all of the DNS names and/or Node IDs being hosted or by several certificates each authenticating a single DNS name and/or Node ID, using the SNI value from the peer to select which certificate to use. The logic for mapping an SNI DNS name to an end-entity certificate is an implementation matter, and can involve correlating DNS name with Node ID or other certificate attributes.

3.5. Session Keeping Policies

This specification gives requirements about how to initiate, sustain, and terminate a TCPCL session but does not impose any requirements on how sessions need to be managed by a BP agent. It is a network administration matter to determine an appropriate session keeping policy, but guidance given here can be used to steer policy toward performance goals.

Persistent Session: This policy preemptively establishes a single session to known entities in the network and keeps the session active using `KEEPALIVES`. Benefits of this policy include reducing the total amount of TCP data needing to be exchanged for a set of transfers (assuming `KEEPALIVE` size is significantly smaller than transfer size), and allowing the session state to indicate peer connectivity. Drawbacks include wasted network resources when a session is mostly idle or when the network connectivity is inconsistent (which requires re-establishing failed sessions), and potential queueing issues when multiple transfers are requested simultaneously. This policy assumes that there is agreement between pairs of entities as to which of the peers will initiate sessions; if there is no such agreement, there is potential for duplicate sessions to be established between peers.

Ephemeral Sessions: This policy only establishes a session when an

outgoing transfer is needed to be sent. Benefits of this policy include not wasting network resources on sessions which are idle for long periods of time, and avoids queueing issues of a persistent session. Drawbacks include the TCP and TLS overhead of establish a new session for each transfer. This policy assumes that each entity can function in a passive role to listen for session requests from any peer which needs to send a transfer; when that is not the case the Polling behavior below needs to happen. This policy can be augmented to keep the session established as long as any transfers are queued.

Active-Only Polling Sessions: When naming and/or addressing of one entity is variable (i.e. dynamically assigned IP address or domain name) or when firewall or routing rules prevent incoming TCP connections, that entity can only function in the active role. In these cases, sessions also need to be established when an incoming transfer is expected from a peer or based on a periodic schedule. This polling behavior causes inefficiencies compared to as-needed ephemeral sessions.

Many other policies can be established in a TCPCL network between the two extremes of single persistent sessions and only ephemeral sessions. Different policies can be applied to each peer entity and to each bundle as it needs to be transferred (e.g for quality of service). Additionally, future session extension types can apply further nuance to session policies and policy negotiation.

3.6. Transfer Segmentation Policies

Each TCPCL session allows a negotiated transfer segmentation policy to be applied in each transfer direction. A receiving entity can set the Segment MRU in its SESS_INIT message to determine the largest acceptable segment size, and a transmitting entity can segment a transfer into any sizes smaller than the receiver's Segment MRU. It is a network administration matter to determine an appropriate segmentation policy for entities operating TCPCL, but guidance given here can be used to steer policy toward performance goals. It is also advised to consider the Segment MRU in relation to chunking/packetization performed by TLS, TCP, and any intermediate network-layer nodes.

Minimum Overhead: For a simple network expected to exchange relatively small bundles, the Segment MRU can be set to be identical to the Transfer MRU which indicates that all transfers can be sent with a single data segment (i.e., no actual segmentation). If the network is closed and all transmitters are known to follow a single-segment transfer policy, then receivers can avoid the necessity of segment reassembly. Because this CL

operates over a TCP stream, which suffers from a form of head-of-queue blocking between messages, while one entity is transmitting a single XFER_SEGMENT message it is not able to transmit any XFER_ACK or XFER_REFUSE for any associated received transfers.

Predictable Message Sizing: In situations where the maximum message size is desired to be well-controlled, the Segment MRU can be set to the largest acceptable size (the message size less XFER_SEGMENT header size) and transmitters can always segment a transfer into maximum-size chunks no larger than the Segment MRU. This guarantees that any single XFER_SEGMENT will not monopolize the TCP stream for too long, which would prevent outgoing XFER_ACK and XFER_REFUSE associated with received transfers.

Dynamic Segmentation: Even after negotiation of a Segment MRU for each receiving entity, the actual transfer segmentation only needs to guarantee that any individual segment is no larger than that MRU. In a situation where TCP throughput is dynamic, the transfer segmentation size can also be dynamic in order to control message transmission duration.

Many other policies can be established in a TCPCL network between the two extremes of minimum overhead (large MRU, single-segment) and predictable message sizing (small MRU, highly segmented). Different policies can be applied to each transfer stream to and from any particular entity. Additionally, future session extension and transfer extension types can apply further nuance to transfer policies and policy negotiation.

3.7. Example Message Exchange

The following figure depicts the protocol exchange for a simple session, showing the session establishment and the transmission of a single bundle split into three data segments (of lengths "L1", "L2", and "L3") from Entity A to Entity B.

Note that the sending entity can transmit multiple XFER_SEGMENT messages without waiting for the corresponding XFER_ACK responses. This enables pipelining of messages on a transfer stream. Although this example only demonstrates a single bundle transmission, it is also possible to pipeline multiple XFER_SEGMENT messages for different bundles without necessarily waiting for XFER_ACK messages to be returned for each one. However, interleaving data segments from different bundles is not allowed.

No errors or rejections are shown in this example.

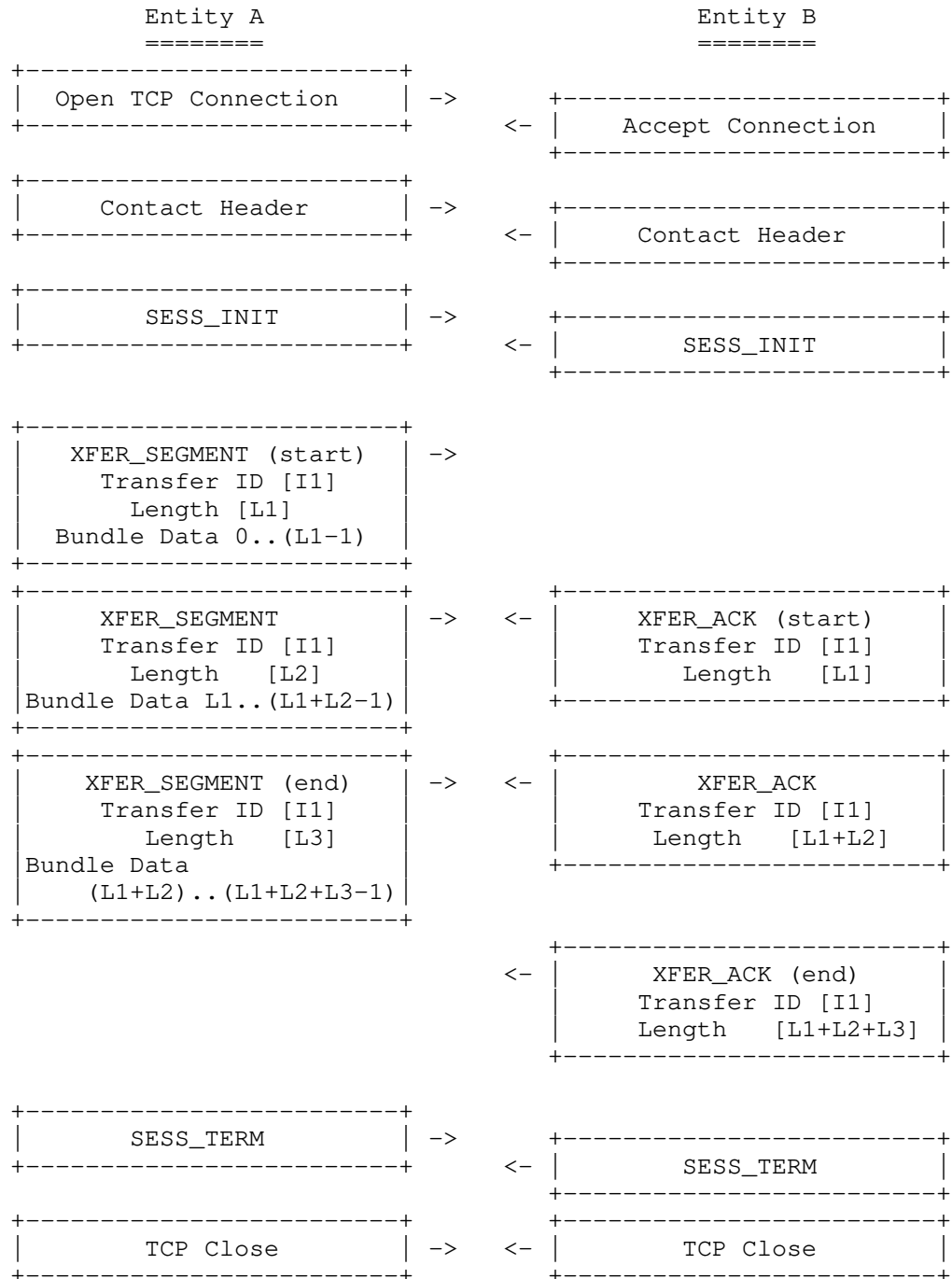


Figure 15: An example of the flow of protocol messages on a single TCP Session between two entities

4. Session Establishment

For bundle transmissions to occur using the TCPCL, a TCPCL session MUST first be established between communicating entities. It is up to the implementation to decide how and when session setup is triggered. For example, some sessions can be opened proactively and maintained for as long as is possible given the network conditions, while other sessions are opened only when there is a bundle that is queued for transmission and the routing algorithm selects a certain next-hop node.

4.1. TCP Connection

To establish a TCPCL session, an entity MUST first establish a TCP connection with the intended peer entity, typically by using the services provided by the operating system. Destination port number 4556 has been assigned by IANA as the Registered Port number for the TCP convergence layer. Other destination port numbers MAY be used per local configuration. Determining a peer's destination port number (if different from the registered TCPCL port number) is up to the implementation. Any source port number MAY be used for TCPCL sessions. Typically an operating system assigned number in the TCP Ephemeral range (49152-65535) is used.

If the entity is unable to establish a TCP connection for any reason, then it is an implementation matter to determine how to handle the connection failure. An entity MAY decide to re-attempt to establish the connection. If it does so, it MUST NOT overwhelm its target with repeated connection attempts. Therefore, the entity MUST NOT retry the connection setup earlier than some delay time from the last attempt, and it SHOULD use a (binary) exponential back-off mechanism to increase this delay in case of repeated failures. The upper limit on a re-attempt back-off is implementation defined but SHOULD be no longer than one minute (60 seconds) before signaling to the BP agent that a connection cannot be made.

Once a TCP connection is established, the active entity SHALL immediately transmit its Contact Header. Once a TCP connection is established, the passive entity SHALL wait for the peer's Contact Header. If the passive entity does not receive a Contact Header after some implementation-defined time duration after TCP connection is established, the entity SHALL close the TCP connection. Entities SHOULD choose a Contact Header reception timeout interval no longer than one minute (60 seconds). Upon reception of a Contact Header, the passive entity SHALL transmit its Contact Header. The ordering

of the Contact Header exchange allows the passive entity to avoid allocating resources to a potential TCPCL session until after a valid Contact Header has been received from the active entity. This ordering also allows the passive peer to adapt to alternate TCPCL protocol versions.

The format of the Contact Header is described in Section 4.2. Because the TCPCL protocol version in use is part of the initial Contact Header, entities using TCPCL version 4 can coexist on a network with entities using earlier TCPCL versions (with some negotiation needed for interoperation as described in Section 4.3).

Within this specification when an entity is said to "close" a TCP connection the entity SHALL use the TCP FIN mechanism and not the RST mechanism. Either mechanism, however, when received will cause a TCP connection to become closed.

4.2. Contact Header

This section describes the format of the Contact Header and the meaning of its fields.

If the entity is configured to enable exchanging messages according to TLS 1.3 [RFC8446] or any successors which are compatible with that TLS ClientHello, the the CAN_TLS flag within its Contact Header SHALL be set to 1. The RECOMMENDED policy is to enable TLS for all sessions, even if security policy does not allow or require authentication. This follows the opportunistic security model of [RFC7435], though an active attacker could interfere with the exchange in such cases (see Section 8.4).

Upon receipt of the Contact Header, both entities perform the validation and negotiation procedures defined in Section 4.3. After receiving the Contact Header from the other entity, either entity MAY refuse the session by sending a SESS_TERM message with an appropriate reason code.

The format for the Contact Header is as follows:

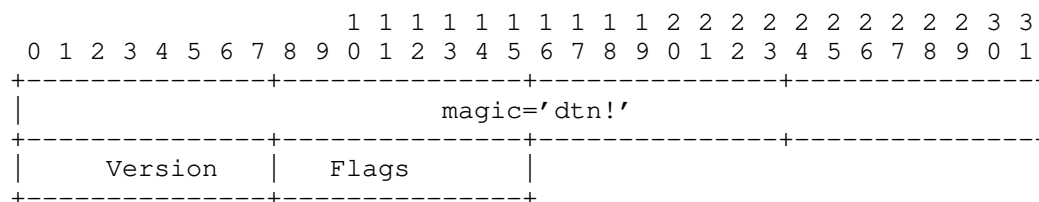


Figure 16: Contact Header Format

See Section 4.3 for details on the use of each of these Contact Header fields.

The fields of the Contact Header are:

magic: A four-octet field that always contains the octet sequence 0x64 0x74 0x6E 0x21, i.e., the text string "dtn!" in US-ASCII (and UTF-8).

Version: A one-octet field value containing the value 4 (current version of the TCPCL).

Flags: A one-octet field of single-bit flags, interpreted according to the descriptions in Table 1. All reserved header flag bits SHALL be set to 0 by the sender. All reserved header flag bits SHALL be ignored by the receiver.

Name	Code	Description
CAN_TLS	0x01	If bit is set, indicates that the sending peer has enabled TLS security.
Reserved	others	

Table 1: Contact Header Flags

4.3. Contact Validation and Negotiation

Upon reception of the Contact Header, each entity follows the following procedures to ensure the validity of the TCPCL session and to negotiate values for the session parameters.

If the magic string is not present or is not valid, the connection MUST be terminated. The intent of the magic string is to provide some protection against an inadvertent TCP connection by a different protocol than the one described in this document. To prevent a flood of repeated connections from a misconfigured application, a passive entity MAY deny new TCP connections from a specific peer address for a period of time after one or more connections fail to provide a decodable Contact Header.

The first negotiation is on the TCPCL protocol version to use. The active entity always sends its Contact Header first and waits for a response from the passive entity. During contact initiation, the active TCPCL entity SHALL send the highest TCPCL protocol version on a first session attempt for a TCPCL peer. If the active entity

receives a Contact Header with a lower protocol version than the one sent earlier on the TCP connection, the TCP connection SHALL be closed. If the active entity receives a SESS_TERM message with reason of "Version Mismatch", that entity MAY attempt further TCPCL sessions with the peer using earlier protocol version numbers in decreasing order. Managing multi-TCPCL-session state such as this is an implementation matter.

If the passive entity receives a Contact Header containing a version that is not a version of the TCPCL that the entity implements, then the entity SHALL send its Contact Header and immediately terminate the session with a reason code of "Version mismatch". If the passive entity receives a Contact Header with a version that is lower than the latest version of the protocol that the entity implements, the entity MAY either terminate the session (with a reason code of "Version mismatch") or adapt its operation to conform to the older version of the protocol. The decision of version fall-back is an implementation matter.

The negotiated contact parameters defined by this specification are described in the following paragraphs.

TCPCL Version: Both Contact Headers of a successful contact negotiation have identical TCPCL Version numbers as described above. Only upon response of a Contact Header from the passive entity is the TCPCL protocol version established and session negotiation begun.

Enable TLS: Negotiation of the Enable TLS parameter is performed by taking the logical AND of the two Contact Headers' CAN_TLS flags. A local security policy is then applied to determine if the negotiated value of Enable TLS is acceptable. It can be a reasonable security policy to require or disallow the use of TLS depending upon the desired network flows. The RECOMMENDED policy is to require TLS for all sessions, even if security policy does not allow or require authentication. Because this state is negotiated over an unsecured medium, there is a risk of a TLS Stripping as described in Section 8.4.

If the Enable TLS state is unacceptable, the entity SHALL terminate the session with a reason code of "Contact Failure". Note that this contact failure reason is different than a failure of TLS handshake or TLS authentication after an agreed-upon and acceptable Enable TLS state. If the negotiated Enable TLS value is true and acceptable then TLS negotiation feature (described in Section 4.4) begins immediately following the Contact Header exchange.

4.4. Session Security

This version of the TCPCL supports establishing a Transport Layer Security (TLS) session within an existing TCP connection. When TLS is used within the TCPCL it affects the entire session. Once TLS is established, there is no mechanism available to downgrade the TCPCL session to non-TLS operation.

Once established, the lifetime of a TLS connection SHALL be bound to the lifetime of the underlying TCP connection. Immediately prior to actively ending a TLS connection after TCPCL session termination, the peer which sent the original (non-reply) SESS_TERM message SHOULD follow the Closure Alert procedure of [RFC8446] to cleanly terminate the TLS connection. Because each TCPCL message is either fixed-length or self-indicates its length, the lack of a TLS Closure Alert will not cause data truncation or corruption.

Subsequent TCPCL session attempts to the same passive entity MAY attempt to use the TLS session resumption feature. There is no guarantee that the passive entity will accept the request to resume a TLS session, and the active entity cannot assume any resumption outcome.

4.4.1. Entity Identification

The TCPCL uses TLS for certificate exchange in both directions to identify each entity and to allow each entity to authenticate its peer. Each certificate can potentially identify multiple entities and there is no problem using such a certificate as long as the identifiers are sufficient to meet authentication policy (as described in later sections) for the entity which presents it.

Because the PKIX environment of each TCPCL entity are likely not controlled by the certificate end users (see Section 3.4), the TCPCL defines a prioritized list of what a certificate can identify about a TCPCL entity:

Node ID: The ideal certificate identity is the Node ID of the entity using the NODE-ID definition below. When the Node ID is identified, there is no need for any lower-level identification to be present (though it can still be present, and if so it is also validated).

DNS Name: If CA policy forbids a certificate to contain an arbitrary

NODE-ID but allows a DNS-ID to be identified then one or more stable DNS names can be identified in the certificate. The use of wildcard DNS-ID is discouraged due to the complex rules for matching and dependence on implementation support for wildcard matching (see Section 6.4.3 of [RFC6125]).

Network Address: If no stable DNS name is available but a stable network address is available and CA policy allows a certificate to contain a IPADDR-ID (as defined below) then one or more network addresses can be identified in the certificate.

This specification defines a NODE-ID of a certificate as being the subjectAltName entry of type otherName with a name form of BundleEID (see Section 4.4.2.1) and a value limited to a Node ID. An entity SHALL ignore any otherName with a name form of BundleEID and a value which is some URI other than a Node ID. The NODE-ID is similar to the URI-ID of [RFC6125] but restricted to a Node ID rather than a URI with a qualified-name authority part. Unless specified otherwise by the definition of the URI scheme being authenticated, URI matching of a NODE-ID SHALL use the URI comparison logic of [RFC3986] and scheme-based normalization of those schemes specified in [I-D.ietf-dtn-bpbis]. A URI scheme can refine this "exact match" logic with rules about how Node IDs within that scheme are to be compared with the certificate-authenticated NODE-ID.

This specification reuses the DNS-ID definition of Section 1.8 of [RFC6125], which is the subjectAltName entry of type dNSName whose value is encoded according to [RFC5280].

This specification defines a IPADDR-ID of a certificate as being the subjectAltName entry of type iPAddress whose value is encoded according to [RFC5280].

4.4.2. Certificate Profile for TCPCL

All end-entity certificates used by a TCPCL entity SHALL conform to [RFC5280], or any updates or successors to that profile. When an end-entity certificate is supplied, the full certification chain SHOULD be included unless security policy indicates that is unnecessary. An entity SHOULD omit the root CA certificate (the last item of the chain) when sending a certification chain, as the recipient already has the root CA to anchor its validation.

The TCPCL requires Version 3 certificates due to the extensions used by this profile. TCPCL entities SHALL reject as invalid Version 1 and Version 2 end-entity certificates.

TCPCL entities SHALL accept certificates that contain an empty Subject field or contain a Subject without a Common Name. Identity information in end-entity certificates is contained entirely in the subjectAltName extension as defined in Section 4.4.1 and below.

All end-entity and CA certificates used for TCPCL SHOULD contain both a Subject Key Identifier and an Authority Key Identifier extension in accordance with [RFC5280]. TCPCL entities SHOULD NOT rely on either a Subject Key Identifier and an Authority Key Identifier being present in any received certificate. Including key identifiers simplifies the work of an entity needing to assemble a certification chain.

Unless prohibited by CA policy, a TCPCL end-entity certificate SHALL contain a NODE-ID which authenticates the Node ID of the peer. When assigned one or more stable DNS names, a TCPCL end-entity certificate SHOULD contain DNS-ID which authenticates those (fully qualified) names. When assigned one or more stable network addresses, a TCPCL end-entity certificate MAY contain IPADDR-ID which authenticates those addresses.

When allowed by CA policy, a BPSec end-entity certificate SHOULD contain a PKIX Extended Key Usage extension in accordance with Section 4.2.1.12 of [RFC5280]. When the PKIX Extended Key Usage extension is present, it SHOULD contain a key purpose id-kp-bundleSecurity (see Section 4.4.2.1). Although not specifically required by TCPCL, some networks or TLS implementations assume the use of id-kp-clientAuth and id-kp-serverAuth are needed for, respectively, the client-side and server-side of TLS authentication. For interoperability, a TCPCL end-entity certificate MAY contain an Extended Key Usage with both id-kp-clientAuth and id-kp-serverAuth values.

When allowed by CA policy, a TCPCL end-entity certificate SHOULD contain a PKIX Key Usage extension in accordance with Section 4.2.1.3 of [RFC5280]. The PKIX Key Usage bit which is consistent with TCPCL security using TLS 1.3 is digitalSignature. The specific algorithms used during the TLS handshake will determine which of those key uses are exercised. Earlier versions of TLS can mandate use of the bits keyEncipherment or keyAgreement.

When allowed by CA policy, a TCPCL end-entity certificate SHOULD contain an Online Certificate Status Protocol (OCSP) URI within an Authority Information Access extension in accordance with Section 4.2.2.1 of [RFC5280].

4.4.2.1. PKIX OID Allocations

This document defines a PKIX Other Name Form identifier of id-on-bundleEID in Appendix B which can be used as the type-id in a subjectAltName entry of type otherName. The BundleEID value associated with otherName type-id id-on-bundleEID SHALL be a URI, encoded as an IA5String, with a scheme which is present in the IANA "Bundle Protocol URI Scheme Type" registry [IANA-BUNDLE]. Although this otherName form allows any Endpoint ID to be present, the NODE-ID defined in Section 4.4.1 limits its use to contain only a Node ID.

This document defines a PKIX Extended Key Usage key purpose id-kp-bundleSecurity in Appendix B which can be used to restrict a certificate's use. The id-kp-bundleSecurity purpose can be combined with other purposes in the same certificate.

4.4.3. TLS Handshake

The use of TLS is negotiated using the Contact Header as described in Section 4.3. After negotiating an Enable TLS parameter of true, and before any other TCPCL messages are sent within the session, the session entities SHALL begin a TLS handshake in accordance with [RFC8446]. By convention, this protocol uses the entity which initiated the underlying TCP connection (the active peer) as the "client" role of the TLS handshake request.

The TLS handshake, if it occurs, is considered to be part of the contact negotiation before the TCPCL session itself is established. Specifics about sensitive data exposure are discussed in Section 8.

The parameters within each TLS negotiation are implementation dependent but any TCPCL entity SHALL follow all recommended practices of BCP 195 [RFC7525], or any updates or successors that become part of BCP 195. Within each TLS handshake, the following requirements apply (using the rough order in which they occur):

Client Hello: When a resolved DNS name was used to establish the TCP connection, the TLS ClientHello SHOULD include a "server_name" extension in accordance with [RFC6066]. When present, the "server_name" extension SHALL contain a "HostName" value taken from the DNS name (of the passive entity) which was resolved. Note: The "HostName" in the "server_name" extension is the network name for the passive entity, not the Node ID of that entity.

Server Certificate: The passive entity SHALL supply a certificate

within the TLS handshake to allow authentication of its side of the session. The supplied end-entity certificate SHALL conform to the profile of Section 4.4.2. The passive entity MAY use the SNI DNS name to choose an appropriate server-side certificate which authenticates that DNS name.

Certificate Request: During TLS handshake, the passive entity SHALL request a client-side certificate.

Client Certificate: The active entity SHALL supply a certificate chain within the TLS handshake to allow authentication of its side of the session. The supplied end-entity certificate SHALL conform to the profile of Section 4.4.2.

If a TLS handshake cannot negotiate a TLS connection, both entities of the TCPCL session SHALL close the TCP connection. At this point the TCPCL session has not yet been established so there is no TCPCL session to terminate.

After a TLS connection is successfully established, the active entity SHALL send a SESS_INIT message to begin session negotiation. This session negotiation and all subsequent messaging are secured.

4.4.4. TLS Authentication

Using PKIX certificates exchanged during the TLS handshake, each of the entities can authenticate a peer Node ID directly or authenticate the peer DNS name or network address. The logic for handling certificates and certificate data is separated into the following phases:

1. Validating the certification path from the end-entity certificate up to a trusted root CA.
2. Validating the Extended Key Usage (EKU) and other properties of the end-entity certificate.
3. Authenticating identities from a valid end-entity certificate.
4. Applying security policy to the result of each identity type authentication.

The result of validating a peer identity (see Section 4.4.1) against one or more type of certificate claim is one of the following:

Absent: Indicating that no such claims are present in the certificate and the identity cannot be authenticated.

Success: Indicating that one or more such claims are present and at least one matches the peer identity value.

Failure: Indicating that one or more such claims are present and none match the peer identity.

4.4.4.1. Certificate Path and Purpose Validation

For any peer end-entity certificate received during TLS handshake, the entity SHALL perform the certification path validation of [RFC5280] up to one of the entity's trusted CA certificates. If enabled by local policy, the entity SHALL perform an OCSP check of each certificate providing OCSP authority information in accordance with [RFC6960]. If certificate validation fails or if security policy disallows a certificate for any reason, the entity SHALL fail the TLS handshake with a "bad_certificate" alert. Leaving out part of the certification chain can cause the entity to fail to validate a certificate if the left-out certificates are unknown to the entity (see Section 8.6).

For the end-entity peer certificate received during TLS handshake, the entity SHALL apply security policy to the Key Usage extension (if present) and Extended Key Usage extension (if present) in accordance with Section 4.2.1.12 of [RFC5280] and the profile in Section 4.4.2.

4.4.4.2. Network-Level Authentication

Either during or immediately after the TLS handshake, if required by security policy each entity SHALL validate the following certificate identifiers together in accordance with Section 6 of [RFC6125]:

- * If the active entity resolved a DNS name (of the passive entity) in order to initiate the TCP connection that DNS name SHALL be used as a DNS-ID reference identifier.
- * The IP address of the other side of the TCP connection SHALL be used as an IPADDR-ID reference identifier.

If the network-level identifiers authentication result is Failure or if the result is Absent and security policy requires an authenticated network-level identifier, the entity SHALL terminate the session (with a reason code of "Contact Failure").

4.4.4.3. Node ID Authentication

Immediately before Session Parameter Negotiation, if required by security policy each entity SHALL validate the certificate NODE-ID in accordance with Section 6 of [RFC6125] using the Node ID of the peer's SESS_INIT message as the NODE-ID reference identifier. If the NODE-ID validation result is Failure or if the result is Absent and security policy requires an authenticated Node ID, the entity SHALL terminate the session (with a reason code of "Contact Failure").

4.4.5. Policy Recommendations

A RECOMMENDED security policy is to enable the use of OCSP checking during TLS handshake. A RECOMMENDED security policy is that if an Extended Key Usage is present that it needs to contain id-kp-bundleSecurity (of Section 4.4.2.1) to be usable with TCPCL security. A RECOMMENDED security policy is to require a validated Node ID (of Section 4.4.4.3) and to ignore any network-level identifier (of Section 4.4.4.2).

This policy relies on and informs the certificate requirements in Section 4.4.3. This policy assumes that a DTN-aware CA (see Section 3.4) will only issue a certificate for a Node ID when it has verified that the private key holder actually controls the DTN node; this is needed to avoid the threat identified in Section 8.9. This policy requires that a certificate contain a NODE-ID and allows the certificate to also contain network-level identifiers. A tailored policy on a more controlled network could relax the requirement on Node ID validation and allow just network-level identifiers to authenticate a peer.

4.4.6. Example TLS Initiation

A summary of a typical TLS use is shown in the sequence in Figure 17 below. In this example the active peer terminates the session but termination can be initiated from either peer.

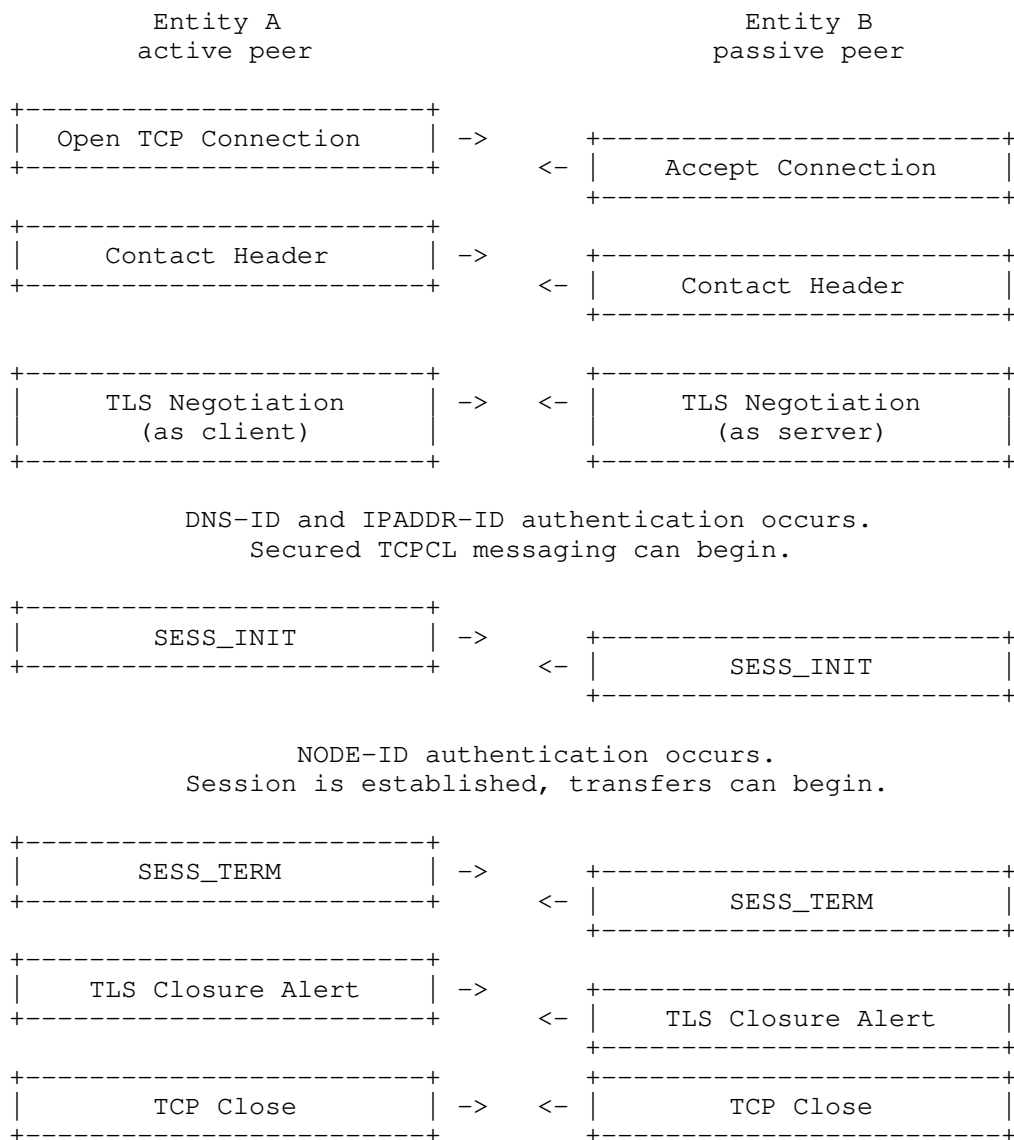


Figure 17: A simple visual example of TCPCL TLS Establishment between two entities

4.5. Message Header

After the initial exchange of a Contact Header and (if TLS is negotiated to be used) the TLS handshake, all messages transmitted over the session are identified by a one-octet header with the following structure:

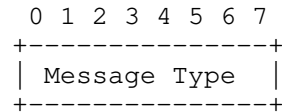


Figure 18: Format of the Message Header

The message header fields are as follows:

Message Type: Indicates the type of the message as per Table 2 below. Encoded values are listed in Section 9.5.

Name	Code	Description
SESS_INIT	0x07	Contains the session parameter inputs from one of the entities, as described in Section 4.6.
SESS_TERM	0x05	Indicates that one of the entities participating in the session wishes to cleanly terminate the session, as described in Section 6.1.
XFER_SEGMENT	0x01	Indicates the transmission of a segment of bundle data, as described in Section 5.2.2.
XFER_ACK	0x02	Acknowledges reception of a data segment, as described in Section 5.2.3.
XFER_REFUSE	0x03	Indicates that the transmission of the current bundle SHALL be stopped, as described in Section 5.2.4.
KEEPALIVE	0x04	Used to keep TCPCL session active, as described in Section 5.1.1.
MSG_REJECT	0x06	Contains a TCPCL message rejection, as described in Section 5.1.2.

Table 2: TCPCL Message Types

4.6. Session Initialization Message (SESS_INIT)

Before a session is established and ready to transfer bundles, the session parameters are negotiated between the connected entities. The SESS_INIT message is used to convey the per-entity parameters which are used together to negotiate the per-session parameters as described in Section 4.7.

The format of a SESS_INIT message is as follows in Figure 19.

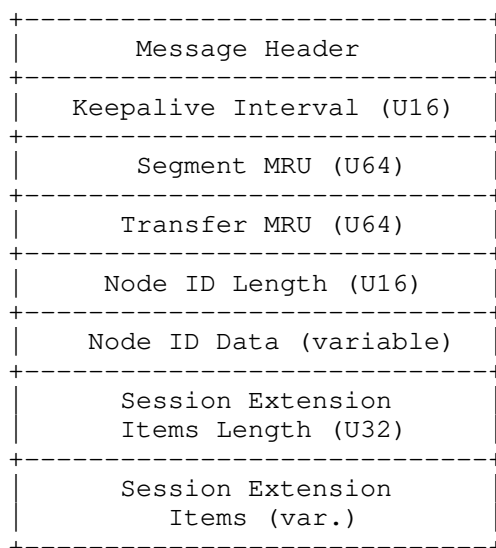


Figure 19: SESS_INIT Format

The fields of the SESS_INIT message are:

Keepalive Interval: A 16-bit unsigned integer indicating the minimum interval, in seconds, to negotiate as the Session Keepalive using the method of Section 4.7.

Segment MRU: A 64-bit unsigned integer indicating the largest allowable single-segment data payload size to be received in this session. Any XFER_SEGMENT sent to this peer SHALL have a data payload no longer than the peer's Segment MRU. The two entities of a single session MAY have different Segment MRUs, and no relation between the two is required.

Transfer MRU: A 64-bit unsigned integer indicating the largest allowable total-bundle data size to be received in this session. Any bundle transfer sent to this peer SHALL have a Total Bundle Length payload no longer than the peer's Transfer MRU. This value can be used to perform proactive bundle fragmentation. The two entities of a single session MAY have different Transfer MRUs, and no relation between the two is required.

Node ID Length and Node ID Data: Together these fields represent a variable-length text string. The Node ID Length is a 16-bit unsigned integer indicating the number of octets of Node ID Data to follow. A zero-length Node ID SHALL be used to indicate the lack of Node ID rather than a truly empty Node ID. This case

allows an entity to avoid exposing Node ID information on an untrusted network. A non-zero-length Node ID Data SHALL contain the UTF-8 encoded Node ID of the Entity which sent the SESS_INIT message. Every Node ID SHALL be a URI consistent with the requirements of [RFC3986] and the URI schemes of the IANA "Bundle Protocol URI Scheme Type" registry [IANA-BUNDLE]. The Node ID itself can be authenticated as described in Section 4.4.4.

Session Extension Length and Session Extension Items: Together these fields represent protocol extension data not defined by this specification. The Session Extension Length is the total number of octets to follow which are used to encode the Session Extension Item list. The encoding of each Session Extension Item is within a consistent data container as described in Section 4.8. The full set of Session Extension Items apply for the duration of the TCPCL session to follow. The order and multiplicity of these Session Extension Items is significant, as defined in the associated type specification(s). If the content of the Session Extension Items data disagrees with the Session Extension Length (e.g., the last Item claims to use more octets than are present in the Session Extension Length), the reception of the SESS_INIT is considered to have failed.

If an entity receives a peer Node ID which is not authenticated (by the procedure of Section 4.4.4.3) that Node ID SHOULD NOT be used by a BP agent for any discovery or routing functions. Trusting an unauthenticated Node ID can lead to the threat described in Section 8.9.

When the active entity initiates a TCPCL session, it is likely based on routing information which binds a Node ID to CL parameters used to initiate the session. If the active entity receives a SESS_INIT with different Node ID than was intended for the TCPCL session, the session MAY be allowed to be established. If allowed, such a session SHALL be associated with the Node ID provided in the SESS_INIT message rather than any intended value.

4.7. Session Parameter Negotiation

An entity calculates the parameters for a TCPCL session by negotiating the values from its own preferences (conveyed by the SESS_INIT it sent to the peer) with the preferences of the peer entity (expressed in the SESS_INIT that it received from the peer). The negotiated parameters defined by this specification are described in the following paragraphs.

Transfer MTU and Segment MTU: The maximum transmit unit (MTU) for

whole transfers and individual segments are identical to the Transfer MRU and Segment MRU, respectively, of the received SESS_INIT message. A transmitting peer can send individual segments with any size smaller than the Segment MTU, depending on local policy, dynamic network conditions, etc. Determining the size of each transmitted segment is an implementation matter. If either the Transfer MRU or Segment MRU is unacceptable, the entity SHALL terminate the session with a reason code of "Contact Failure".

Session Keepalive: Negotiation of the Session Keepalive parameter is performed by taking the minimum of the two Keepalive Interval values from the two SESS_INIT messages. The Session Keepalive interval is a parameter for the behavior described in Section 5.1.1. If the Session Keepalive interval is unacceptable, the entity SHALL terminate the session with a reason code of "Contact Failure". Note: a negotiated Session Keepalive of zero indicates that KEEPALIVES are disabled.

Once this process of parameter negotiation is completed, this protocol defines no additional mechanism to change the parameters of an established session; to effect such a change, the TCPCL session MUST be terminated and a new session established.

4.8. Session Extension Items

Each of the Session Extension Items SHALL be encoded in an identical Type-Length-Value (TLV) container form as indicated in Figure 20.

The fields of the Session Extension Item are:

Item Flags: A one-octet field containing generic bit flags about the Item, which are listed in Table 3. All reserved header flag bits SHALL be set to 0 by the sender. All reserved header flag bits SHALL be ignored by the receiver. If a TCPCL entity receives a Session Extension Item with an unknown Item Type and the CRITICAL flag of 1, the entity SHALL terminate the TCPCL session with SESS_TERM reason code of "Contact Failure". If the CRITICAL flag is 0, an entity SHALL skip over and ignore any item with an unknown Item Type.

Item Type: A 16-bit unsigned integer field containing the type of the extension item. This specification does not define any extension types directly, but does create an IANA registry for such codes (see Section 9.3).

Item Length: A 16-bit unsigned integer field containing the number of Item Value octets to follow.

Item Value: A variable-length data field which is interpreted according to the associated Item Type. This specification places no restrictions on an extension's use of available Item Value data. Extension specifications SHOULD avoid the use of large data lengths, as no bundle transfers can begin until the full extension data is sent.

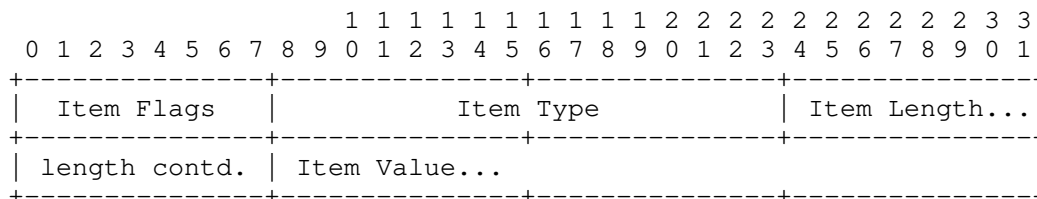


Figure 20: Session Extension Item Format

Name	Code	Description
CRITICAL	0x01	If bit is set, indicates that the receiving peer must handle the extension item.
Reserved	others	

Table 3: Session Extension Item Flags

5. Established Session Operation

This section describes the protocol operation for the duration of an established session, including the mechanism for transmitting bundles over the session.

5.1. Upkeep and Status Messages

5.1.1. Session Upkeep (KEEPALIVE)

The protocol includes a provision for transmission of KEEPALIVE messages over the TCPCL session to help determine if the underlying TCP connection has been disrupted.

As described in Section 4.3, a negotiated parameter of each session is the Session Keepalive interval. If the negotiated Session Keepalive is zero (i.e., one or both SESS_INIT messages contains a zero Keepalive Interval), then the keepalive feature is disabled. There is no logical minimum value for the keepalive interval (within the minimum imposed by the positive-value encoding), but when used

for many sessions on an open, shared network a short interval could lead to excessive traffic. For shared network use, entities SHOULD choose a keepalive interval no shorter than 30 seconds. There is no logical maximum value for the keepalive interval (within the maximum imposed by the fixed-size encoding), but an idle TCP connection is liable for closure by the host operating system if the keepalive time is longer than tens-of-minutes. Entities SHOULD choose a keepalive interval no longer than 10 minutes (600 seconds).

Note: The Keepalive Interval SHOULD NOT be chosen too short as TCP retransmissions MAY occur in case of packet loss. Those will have to be triggered by a timeout (TCP retransmission timeout (RTO)), which is dependent on the measured RTT for the TCP connection so that KEEPALIVE messages can experience noticeable latency.

The format of a KEEPALIVE message is a one-octet message type code of KEEPALIVE (as described in Table 2) with no additional data. Both sides SHALL send a KEEPALIVE message whenever the negotiated interval has elapsed with no transmission of any message (KEEPALIVE or other).

If no message (KEEPALIVE or other) has been received in a session after some implementation-defined time duration, then the entity SHALL terminate the session by transmitting a SESS_TERM message (as described in Section 6.1) with reason code "Idle Timeout". If configurable, the idle timeout duration SHOULD be no shorter than twice the keepalive interval. If not configurable, the idle timeout duration SHOULD be exactly twice the keepalive interval.

5.1.2. Message Rejection (MSG_REJECT)

This message type is not expected to be seen in a well-functioning session. Its purpose is to aid in troubleshooting bad entity behavior by allowing the peer to observe why an entity is not responding as expected to its messages.

If a TCPCL entity receives a message type which is unknown to it (possibly due to an unhandled protocol version mismatch or a incorrectly-negotiated session extension which defines a new message type), the entity SHALL send a MSG_REJECT message with a Reason Code of "Message Type Unknown" and close the TCP connection. If a TCPCL entity receives a message type which is known but is inappropriate for the negotiated session parameters (possibly due to incorrectly-negotiated session extension), the entity SHALL send a MSG_REJECT message with a Reason Code of "Message Unsupported". If a TCPCL entity receives a message which is inappropriate for the current session state (e.g., a SESS_INIT after the session has already been established or an XFER_ACK message with an unknown Transfer ID), the entity SHALL send a MSG_REJECT message with a Reason Code of "Message Unexpected".

The format of a MSG_REJECT message is as follows in Figure 21.

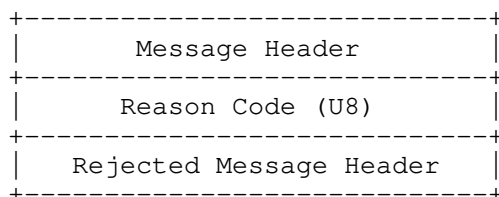


Figure 21: Format of MSG_REJECT Messages

The fields of the MSG_REJECT message are:

Reason Code: A one-octet refusal reason code interpreted according to the descriptions in Table 4.

Rejected Message Header: The Rejected Message Header is a copy of the Message Header to which the MSG_REJECT message is sent as a response.

Name	Code	Description
Message Type Unknown	0x01	A message was received with a Message Type code unknown to the TCPCL entity.
Message Unsupported	0x02	A message was received but the TCPCL entity cannot comply with the message contents.
Message Unexpected	0x03	A message was received while the session is in a state in which the message is not expected.

Table 4: MSG_REJECT Reason Codes

5.2. Bundle Transfer

All of the messages in this section are directly associated with transferring a bundle between TCPCL entities.

A single TCPCL transfer results in a bundle (handled by the convergence layer as opaque data) being exchanged from one entity to the other. In TCPCL a transfer is accomplished by dividing a single bundle up into "segments" based on the receiving-side Segment MRU (see Section 4.2). The choice of the length to use for segments is an implementation matter, but each segment MUST NOT be larger than the receiving entity's maximum receive unit (MRU) (see the field Segment MRU of Section 4.2). The first segment for a bundle is indicated by the 'START' flag and the last segment is indicated by the 'END' flag.

A single transfer (and by extension a single segment) SHALL NOT contain data of more than a single bundle. This requirement is imposed on the agent using the TCPCL rather than TCPCL itself.

If multiple bundles are transmitted on a single TCPCL connection, they MUST be transmitted consecutively without interleaving of segments from multiple bundles.

5.2.1. Bundle Transfer ID

Each of the bundle transfer messages contains a Transfer ID which is used to correlate messages (from both sides of a transfer) for each bundle. A Transfer ID does not attempt to address uniqueness of the bundle data itself and has no relation to concepts such as bundle fragmentation. Each invocation of TCPCL by the bundle protocol agent, requesting transmission of a bundle (fragmentary or otherwise), results in the initiation of a single TCPCL transfer. Each transfer entails the sending of a sequence of some number of XFER_SEGMENT and XFER_ACK messages; all are correlated by the same Transfer ID. The sending entity originates a transfer ID and the receiving entity uses that same Transfer ID in acknowledgements.

Transfer IDs from each entity SHALL be unique within a single TCPCL session. Upon exhaustion of the entire 64-bit Transfer ID space, the sending entity SHALL terminate the session with SESS_TERM reason code "Resource Exhaustion". For bidirectional bundle transfers, a TCPCL entity SHOULD NOT rely on any relation between Transfer IDs originating from each side of the TCPCL session.

Although there is not a strict requirement for Transfer ID initial values or ordering (see Section 8.13), in the absence of any other mechanism for generating Transfer IDs an entity SHALL use the following algorithm: The initial Transfer ID from each entity is zero and subsequent Transfer ID values are incremented from the prior Transfer ID value by one.

5.2.2. Data Transmission (XFER_SEGMENT)

Each bundle is transmitted in one or more data segments. The format of a XFER_SEGMENT message follows in Figure 22.

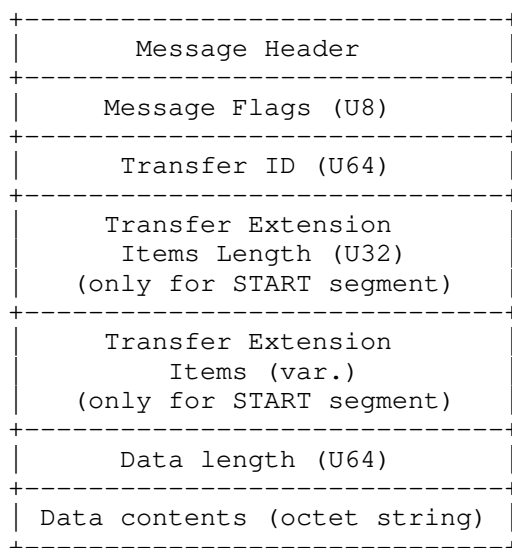


Figure 22: Format of XFER_SEGMENT Messages

The fields of the XFER_SEGMENT message are:

Message Flags: A one-octet field of single-bit flags, interpreted according to the descriptions in Table 5. All reserved header flag bits SHALL be set to 0 by the sender. All reserved header flag bits SHALL be ignored by the receiver.

Transfer ID: A 64-bit unsigned integer identifying the transfer being made.

Transfer Extension Length and Transfer Extension Items: Together these fields represent protocol extension data for this specification. The Transfer Extension Length and Transfer Extension Item fields SHALL only be present when the 'START' flag is set to 1 on the message. The Transfer Extension Length is the total number of octets to follow which are used to encode the Transfer Extension Item list. The encoding of each Transfer Extension Item is within a consistent data container as described in Section 5.2.5. The full set of transfer extension items apply only to the associated single transfer. The order and multiplicity of these transfer extension items is significant, as defined in the associated type specification(s). If the content of the Transfer Extension Items data disagrees with the Transfer Extension Length (e.g., the last Item claims to use more octets than are present in the Transfer Extension Length), the reception of the XFER_SEGMENT is considered to have failed.

Data length: A 64-bit unsigned integer indicating the number of octets in the Data contents to follow.

Data contents: The variable-length data payload of the message.

Name	Code	Description
END	0x01	If bit is set, indicates that this is the last segment of the transfer.
START	0x02	If bit is set, indicates that this is the first segment of the transfer.
Reserved	others	

Table 5: XFER_SEGMENT Flags

The flags portion of the message contains two flag values in the two low-order bits, denoted 'START' and 'END' in Table 5. The 'START' flag SHALL be set to 1 when transmitting the first segment of a transfer. The 'END' flag SHALL be set to 1 when transmitting the last segment of a transfer. In the case where an entire transfer is accomplished in a single segment, both the 'START' and 'END' flags SHALL be set to 1.

Once a transfer of a bundle has commenced, the entity MUST only send segments containing sequential portions of that bundle until it sends a segment with the 'END' flag set to 1. No interleaving of multiple transfers from the same entity is possible within a single TCPCL session. Simultaneous transfers between two entities MAY be achieved using multiple TCPCL sessions.

5.2.3. Data Acknowledgments (XFER_ACK)

Although the TCP transport provides reliable transfer of data between transport peers, the typical BSD sockets interface provides no means to inform a sending application of when the receiving application has processed some amount of transmitted data. Thus, after transmitting some data, the TCPCL needs an additional mechanism to determine whether the receiving agent has successfully received and fully processed the segment. To this end, the TCPCL protocol provides feedback messaging whereby a receiving entity transmits acknowledgments of reception of data segments.

The format of an XFER_ACK message follows in Figure 23.

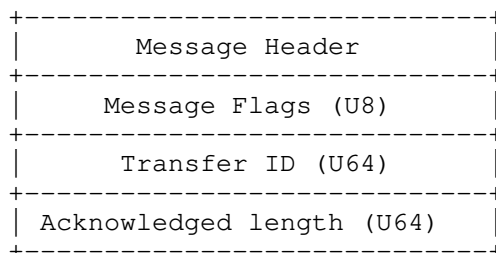


Figure 23: Format of XFER_ACK Messages

The fields of the XFER_ACK message are:

Message Flags: A one-octet field of single-bit flags, interpreted according to the descriptions in Table 5. All reserved header flag bits SHALL be set to 0 by the sender. All reserved header flag bits SHALL be ignored by the receiver.

Transfer ID: A 64-bit unsigned integer identifying the transfer being acknowledged.

Acknowledged length: A 64-bit unsigned integer indicating the total number of octets in the transfer which are being acknowledged.

A receiving TCPCL entity SHALL send an XFER_ACK message in response to each received XFER_SEGMENT message after the segment has been fully processed. The flags portion of the XFER_ACK header SHALL be set to match the corresponding XFER_SEGMENT message being acknowledged (including flags not decodable to the entity). The acknowledged length of each XFER_ACK contains the sum of the data length fields of all XFER_SEGMENT messages received so far in the course of the indicated transfer. The sending entity SHOULD transmit multiple XFER_SEGMENT messages without waiting for the corresponding XFER_ACK responses. This enables pipelining of messages on a transfer stream.

For example, suppose the sending entity transmits four segments of bundle data with lengths 100, 200, 500, and 1000, respectively. After receiving the first segment, the entity sends an acknowledgment of length 100. After the second segment is received, the entity sends an acknowledgment of length 300. The third and fourth acknowledgments are of length 800 and 1800, respectively.

5.2.4. Transfer Refusal (XFER_REFUSE)

The TCPCL supports a mechanism by which a receiving entity can indicate to the sender that it does not want to receive the corresponding bundle. To do so, upon receiving an XFER_SEGMENT message, the entity MAY transmit a XFER_REFUSE message. As data segments and acknowledgments can cross on the wire, the bundle that is being refused SHALL be identified by the Transfer ID of the refusal.

There is no required relation between the Transfer MRU of a TCPCL entity (which is supposed to represent a firm limitation of what the entity will accept) and sending of a XFER_REFUSE message. A XFER_REFUSE can be used in cases where the agent's bundle storage is temporarily depleted or somehow constrained. A XFER_REFUSE can also be used after the bundle header or any bundle data is inspected by an agent and determined to be unacceptable.

A transfer receiver MAY send an XFER_REFUSE message as soon as it receives any XFER_SEGMENT message. The transfer sender MUST be prepared for this and MUST associate the refusal with the correct bundle via the Transfer ID fields.

The TCPCL itself does not have any required behavior to respond to an XFER_REFUSE based on its Reason Code; the refusal is passed up as an indication to the BP agent that the transfer has been refused. If a transfer refusal has a Reason Code which is not decodable to the BP agent, the agent SHOULD treat the refusal as having an Unknown reason.

The format of the XFER_REFUSE message is as follows in Figure 24.

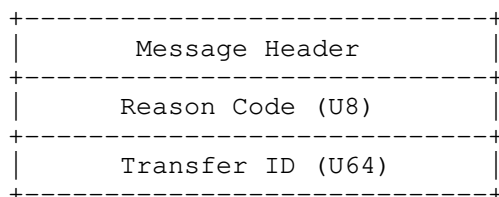


Figure 24: Format of XFER_REFUSE Messages

The fields of the XFER_REFUSE message are:

Reason Code: A one-octet refusal reason code interpreted according to the descriptions in Table 6.

Transfer ID: A 64-bit unsigned integer identifying the transfer

being refused.

Name	Code	Description
Unknown	0x00	Reason for refusal is unknown or not specified.
Completed	0x01	The receiver already has the complete bundle. The sender MAY consider the bundle as completely received.
No Resources	0x02	The receiver's resources are exhausted. The sender SHOULD apply reactive bundle fragmentation before retrying.
Retransmit	0x03	The receiver has encountered a problem that requires the bundle to be retransmitted in its entirety.
Not Acceptable	0x04	Some issue with the bundle data or the transfer extension data was encountered. The sender SHOULD NOT retry the same bundle with the same extensions.
Extension Failure	0x05	A failure processing the Transfer Extension Items has occurred.
Session Terminating	0x06	The receiving entity is in the process of terminating the session. The sender MAY retry the same bundle at a later time in a different session.

Table 6: XFER_REFUSE Reason Codes

The receiver MUST, for each transfer preceding the one to be refused, have either acknowledged all XFER_SEGMENT messages or refused the bundle transfer.

The bundle transfer refusal MAY be sent before an entire data segment is received. If a sender receives a XFER_REFUSE message, the sender MUST complete the transmission of any partially sent XFER_SEGMENT message. There is no way to interrupt an individual TCPCL message partway through sending it. The sender MUST NOT commence transmission of any further segments of the refused bundle subsequently. Note, however, that this requirement does not ensure that an entity will not receive another XFER_SEGMENT for the same

bundle after transmitting a XFER_REFUSE message since messages can cross on the wire; if this happens, subsequent segments of the bundle SHALL also be refused with a XFER_REFUSE message.

Note: If a bundle transmission is aborted in this way, the receiver does not receive a segment with the 'END' flag set to 1 for the aborted bundle. The beginning of the next bundle is identified by the 'START' flag set to 1, indicating the start of a new transfer, and with a distinct Transfer ID value.

5.2.5. Transfer Extension Items

Each of the Transfer Extension Items SHALL be encoded in an identical Type-Length-Value (TLV) container form as indicated in Figure 25.

The fields of the Transfer Extension Item are:

Item Flags: A one-octet field containing generic bit flags about the Item, which are listed in Table 7. All reserved header flag bits SHALL be set to 0 by the sender. All reserved header flag bits SHALL be ignored by the receiver. If a TCPCL entity receives a Transfer Extension Item with an unknown Item Type and the CRITICAL flag is 1, the entity SHALL refuse the transfer with an XFER_REFUSE reason code of "Extension Failure". If the CRITICAL flag is 0, an entity SHALL skip over and ignore any item with an unknown Item Type.

Item Type: A 16-bit unsigned integer field containing the type of the extension item. This specification creates an IANA registry for such codes (see Section 9.4).

Item Length: A 16-bit unsigned integer field containing the number of Item Value octets to follow.

Item Value: A variable-length data field which is interpreted according to the associated Item Type. This specification places no restrictions on an extension's use of available Item Value data. Extension specifications SHOULD avoid the use of large data lengths, as the associated transfer cannot begin until the full extension data is sent.

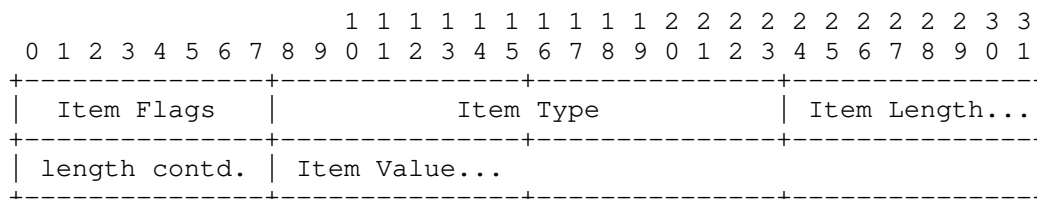


Figure 25: Transfer Extension Item Format

Name	Code	Description
CRITICAL	0x01	If bit is set, indicates that the receiving peer must handle the extension item.
Reserved	others	

Table 7: Transfer Extension Item Flags

5.2.5.1. Transfer Length Extension

The purpose of the Transfer Length extension is to allow entities to preemptively refuse bundles that would exceed their resources or to prepare storage on the receiving entity for the upcoming bundle data.

Multiple Transfer Length extension items SHALL NOT occur within the same transfer. The lack of a Transfer Length extension item in any transfer SHALL NOT imply anything about the potential length of the transfer. The Transfer Length extension SHALL be assigned transfer extension type ID 0x0001.

If a transfer occupies exactly one segment (i.e., both START and END flags are 1) the Transfer Length extension SHOULD NOT be present. The extension does not provide any additional information for single-segment transfers.

The format of the Transfer Length data is as follows in Figure 26.

Total Length (U64)

Figure 26: Format of Transfer Length data

The fields of the Transfer Length extension are:

Total Length: A 64-bit unsigned integer indicating the size of the data-to-be-transferred. The Total Length field SHALL be treated as authoritative by the receiver. If, for whatever reason, the actual total length of bundle data received differs from the value indicated by the Total Length value, the receiver SHALL treat the transmitted data as invalid and send an XFER_REFUSE with a Reason Code of "Not Acceptable".

6. Session Termination

This section describes the procedures for terminating a TCPCL session. The purpose of terminating a session is to allow transfers to complete before the TCP connection is closed but not allow any new transfers to start. A session state change is necessary for this to happen because transfers can be in-progress in either direction (transfer stream) within a session. Waiting for a transfer to complete in one direction does not control or influence the possibility of a transfer in the other direction. Either peer of a session can terminate an established session at any time.

6.1. Session Termination Message (SESS_TERM)

To cleanly terminate a session, a SESS_TERM message SHALL be transmitted by either entity at any point following complete transmission of any other message. When sent to initiate a termination, the REPLY flag of a SESS_TERM message SHALL be 0. Upon receiving a SESS_TERM message after not sending a SESS_TERM message in the same session, an entity SHALL send an acknowledging SESS_TERM message. When sent to acknowledge a termination, a SESS_TERM message SHALL have identical data content from the message being acknowledged except for the REPLY flag, which is set to 1 to indicate acknowledgement.

Once a SESS_TERM message is sent the state of that TCPCL session changes to Ending. While the session is in the Ending state, an entity MAY finish an in-progress transfer in either direction. While the session is in the Ending state, an entity SHALL NOT begin any new outgoing transfer for the remainder of the session. While the session is in the Ending state, an entity SHALL NOT accept any new incoming transfer for the remainder of the session. If a new incoming transfer is attempted while in the Ending state, the receiving entity SHALL send an XFER_REFUSE with a Reason Code of "Session Terminating".

There are circumstances where an entity has an urgent need to close a TCP connection associated with a TCPCL session, without waiting for transfers to complete but also in a way which doesn't force timeouts to occur; for example, due to impending shutdown of the underlying data link layer. Instead of following a clean termination sequence, after transmitting a SESS_TERM message an entity MAY perform an unclean termination by immediately closing the associated TCP connection. When performing an unclean termination, an entity SHOULD acknowledge all received XFER_SEGMENTS with an XFER_ACK before closing the TCP connection. Not acknowledging received segments can result in unnecessary bundle or bundle fragment retransmission. Any delay between request to close the TCP connection and actual closing

of the connection (a "half-closed" state) MAY be ignored by the TCPCL entity. If the underlying TCP connection is closed during a transmission (in either transfer stream), the transfer SHALL be indicated to the BP agent as failed (see the transmission failure and reception failure indications of Section 3.1).

The TCPCL itself does not have any required behavior to respond to an SESS_TERM based on its Reason Code; the termination is passed up as an indication to the BP agent that the session state has changed. If a termination has a Reason Code which is not decodable to the BP agent, the agent SHOULD treat the termination as having an Unknown reason.

The format of the SESS_TERM message is as follows in Figure 27.

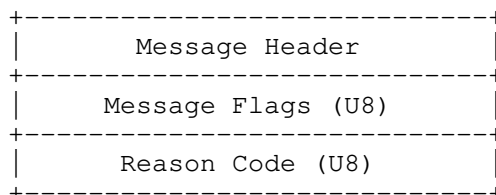


Figure 27: Format of SESS_TERM Messages

The fields of the SESS_TERM message are:

Message Flags: A one-octet field of single-bit flags, interpreted according to the descriptions in Table 8. All reserved header flag bits SHALL be set to 0 by the sender. All reserved header flag bits SHALL be ignored by the receiver.

Reason Code: A one-octet refusal reason code interpreted according to the descriptions in Table 9.

Name	Code	Description
REPLY	0x01	If bit is set, indicates that this message is an acknowledgement of an earlier SESS_TERM message.
Reserved	others	

Table 8: SESS_TERM Flags

Name	Code	Description
Unknown	0x00	A termination reason is not available.
Idle timeout	0x01	The session is being terminated due to idleness.
Version mismatch	0x02	The entity cannot conform to the specified TCPCL protocol version.
Busy	0x03	The entity is too busy to handle the current session.
Contact Failure	0x04	The entity cannot interpret or negotiate a Contact Header or SESS_INIT option.
Resource Exhaustion	0x05	The entity has run into some resource limit and cannot continue the session.

Table 9: SESS_TERM Reason Codes

The earliest a TCPCL session termination MAY occur is immediately after transmission of a Contact Header (and prior to any further message transmit). This can, for example, be used to notify that the entity is currently not able or willing to communicate. However, an entity MUST always send the Contact Header to its peer before sending a SESS_TERM message.

Termination of the TCP connection MAY occur prior to receiving the Contact header as discussed in Section 4.1. If reception of the Contact Header itself somehow fails (e.g., an invalid "magic string" is received), an entity SHALL close the TCP connection without sending a SESS_TERM message.

If a session is to be terminated before a protocol message has completed being sent, then the entity MUST NOT transmit the SESS_TERM message but still SHALL close the TCP connection. Each TCPCL message is contiguous in the octet stream and has no ability to be cut short and/or preempted by an other message. This is particularly important when large segment sizes are being transmitted; either entire XFER_SEGMENT is sent before a SESS_TERM message or the connection is simply terminated mid-XFER_SEGMENT.

6.2. Idle Session Shutdown

The protocol includes a provision for clean termination of idle sessions. Determining the length of time to wait before terminating idle sessions, if they are to be terminated at all, is an implementation and configuration matter.

If there is a configured time to terminate idle sessions and if no TCPCL messages (other than KEEPALIVE messages) has been received for at least that amount of time, then either entity MAY terminate the session by transmitting a SESS_TERM message indicating the reason code of "Idle timeout" (as described in Table 9).

7. Implementation Status

This section is to be removed before publishing as an RFC.

[NOTE to the RFC Editor: please remove this section before publication, as well as the reference to [RFC7942], [github-dtn-demo-agent], and [github-dtn-wireshark].]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations can exist.

An example implementation of the this draft of TCPCLv4 has been created as a GitHub project [github-dtn-demo-agent] and is intended to use as a proof-of-concept and as a possible source of interoperability testing. This example implementation uses D-Bus as the CL-BP Agent interface, so it only runs on hosts which provide the Python "dbus" library.

A wireshark dissector for TCPCLv4 has been created as a GitHub project [github-dtn-wireshark] and has been kept in-sync with the latest encoding of this specification.

8. Security Considerations

This section separates security considerations into threat categories based on guidance of BCP 72 [RFC3552].

8.1. Threat: Passive Leak of Node Data

When used without TLS security, the TCPCL exposes the Node ID and other configuration data to passive eavesdroppers. This occurs even when no transfers occur within a TCPCL session. This can be avoided by always using TLS, even if authentication is not available (see Section 8.12).

8.2. Threat: Passive Leak of Bundle Data

TCPCL can be used to provide point-to-point transport security, but does not provide security of data-at-rest and does not guarantee end-to-end bundle security. The bundle security mechanisms defined in [I-D.ietf-dtn-bpsec] are to be used instead.

When used without TLS security, the TCPCL exposes all bundle data to passive eavesdroppers. This can be avoided by always using TLS, even if authentication is not available (see Section 8.12).

8.3. Threat: TCPCL Version Downgrade

When a TCPCL entity supports multiple versions of the protocol it is possible for a malicious or misconfigured peer to use an older version of TCPCL which does not support transport security. A on-path attacker can also manipulate a Contact Header to present a lower protocol version than desired.

It is up to security policies within each TCPCL entity to ensure that the negotiated TCPCL version meets transport security requirements.

8.4. Threat: Transport Security Stripping

When security policy allows non-TLS sessions, TCPCL does not protect against active network attackers. It is possible for a on-path attacker to set the CAN_TLS flag to 0 on either side of the Contact Header exchange, which will cause the negotiation of Section 4.3 to disable TLS. This leads to the "SSL Stripping" attack described in [RFC7457].

The purpose of the CAN_TLS flag is to allow the use of TCPCL on entities which simply do not have a TLS implementation available. When TLS is available on an entity, it is strongly encouraged that the security policy disallow non-TLS sessions. This requires that the TLS handshake occurs, regardless of the policy-driven parameters of the handshake and policy-driven handling of the handshake outcome.

One mechanism to mitigate the possibility of TLS stripping is the use of DNS-based Authentication of Named Entities (DANE) [RFC6698] toward the passive peer. This mechanism relies on DNS and is unidirectional, so it doesn't help with applying policy toward the active peer, but it can be useful in an environment using opportunistic security. The configuration and use of DANE are outside of the scope of this document.

The negotiated use of TLS is identical behavior to STARTTLS use in [RFC2595], [RFC4511], and others.

8.5. Threat: Weak TLS Configurations

Even when using TLS to secure the TCPCL session, the actual ciphersuite negotiated between the TLS peers can be insecure. Recommendations for ciphersuite use are included in BCP 195 [RFC7525]. It is up to security policies within each TCPCL entity to ensure that the negotiated TLS ciphersuite meets transport security requirements.

8.6. Threat: Untrusted End-Entity Certificate

The profile in Section 4.4.4 uses end-entity certificates chained up to a trusted root CA. During TLS handshake, either entity can send a certificate set which does not contain the full chain, possibly excluding intermediate or root CAs. In an environment where peers are known to already contain needed root and intermediate CAs there is no need to include those CAs, but this has a risk of an entity not actually having one of the needed CAs.

8.7. Threat: Certificate Validation Vulnerabilities

Even when TLS itself is operating properly an attacker can attempt to exploit vulnerabilities within certificate check algorithms or configuration to establish a secure TCPCL session using an invalid certificate. A BP agent treats the peer Node ID within a TCPCL session as authoritative and an invalid certificate exploit could lead to bundle data leaking and/or denial of service to the Node ID being impersonated.

There are many reasons, described in [RFC5280] and [RFC6125], why a certificate can fail to validate, including using the certificate outside of its valid time interval, using purposes for which it was not authorized, or using it after it has been revoked by its CA. Validating a certificate is a complex task and can require network connectivity outside of the primary TCPCL network path(s) if a mechanism such as OCSP [RFC6960] is used by the CA. The configuration and use of particular certificate validation methods are outside of the scope of this document.

8.8. Threat: Symmetric Key Limits

Even with a secure block cipher and securely-established session keys, there are limits to the amount of plaintext which can be safely encrypted with a given set of keys as described in [AEAD-LIMITS]. When permitted by the negotiated TLS version (see [RFC8446]), it is advisable to take advantage of session key updates to avoid those limits.

8.9. Threat: BP Node Impersonation

The certificates exchanged by TLS enable authentication of peer DNS name and Node ID, but it is possible that a peer either not provide a valid certificate or that the certificate does not validate either the DNS-ID/IPADDR-ID or NODE-ID of the peer (see Section 3.4). Having a CA-validated certificate does not alone guarantee the identity of the network host or BP node from which the certificate is provided; additional validation procedures in Section 4.4.3 bind the DNS-ID/IPADDR-ID or NODE-ID based on the contents of the certificate.

The DNS-ID/IPADDR-ID validation is a weaker form of authentication, because even if a peer is operating on an authenticated network DNS name or IP address it can provide an invalid Node ID and cause bundles to be "leaked" to an invalid node. Especially in DTN environments, network names and addresses of nodes can be time-variable so binding a certificate to a Node ID is a more stable identity.

NODE-ID validation ensures that the peer to which a bundle is transferred is in fact the node which the BP Agent expects it to be. In circumstances where certificates can only be issued to DNS names, Node ID validation is not possible but it could be reasonable to assume that a trusted host is not going to present an invalid Node ID. Determining when a DNS-ID/IPADDR-ID authentication can be trusted to validate a Node ID is also a policy matter outside of the scope of this document.

One mitigation to arbitrary entities with valid PKIX certificates impersonating arbitrary Node IDs is the use of the PKIX Extended Key Usage key purpose id-kp-bundleSecurity (see Section 4.4.2.1). When this Extended Key Usage is present in the certificate, it represents a stronger assertion that the private key holder should in fact be trusted to operate as a DTN Node.

8.10. Threat: Denial of Service

The behaviors described in this section all amount to a potential denial-of-service to a TCPCL entity. The denial-of-service could be limited to an individual TCPCL session, could affect other well-behaving sessions on an entity, or could affect all sessions on a host.

A malicious entity can continually establish TCPCL sessions and delay sending of protocol-required data to trigger timeouts. The victim entity can block TCP connections from network peers which are thought to be incorrectly behaving within TCPCL.

An entity can send a large amount of data over a TCPCL session, requiring the receiving entity to handle the data. The victim entity can attempt to stop the flood of data by sending an XFER_REFUSE message, or forcibly terminate the session.

There is the possibility of a "data dribble" attack in which an entity presents a very small Segment MRU which causes transfers to be split among an large number of very small segments and causes the segmentation overhead to overwhelm the actual bundle data segments. Similarly, an entity can present a very small Transfer MRU which will cause resources to be wasted on establishment and upkeep of a TCPCL session over which a bundle could never be transferred. The victim entity can terminate the session during the negotiation of Section 4.7 if the MRUs are unacceptable.

The keepalive mechanism can be abused to waste throughput within a network link which would otherwise be usable for bundle transmissions. Due to the quantization of the Keepalive Interval parameter the smallest Session Keepalive is one second, which should be long enough to not flood the link. The victim entity can terminate the session during the negotiation of Section 4.7 if the Keepalive Interval is unacceptable.

Finally, an attacker or a misconfigured entity can cause issues at the TCP connection which will cause unnecessary TCP retransmissions or connection resets, effectively denying the use of the overlying TCPCL session.

8.11. Mandatory-to-Implement TLS

Following IETF best current practice, TLS is mandatory to implement for all TCPCL implementations but TLS is optional to use for a given TCPCL session. The recommended configuration of Section 4.2 is to always enable TLS, but entities are permitted to disable TLS based on local configuration. The configuration to enable or disable TLS for an entity or a session is outside of the scope of this document. The configuration to disable TLS is different from the threat of TLS stripping described in Section 8.4.

8.12. Alternate Uses of TLS

This specification makes use of PKIX certificate validation and authentication within TLS. There are alternate uses of TLS which are not necessarily incompatible with the security goals of this specification, but are outside of the scope of this document. The following subsections give examples of alternate TLS uses.

8.12.1. TLS Without Authentication

In environments where PKI is available but there are restrictions on the issuance of certificates (including the contents of certificates), it may be possible to make use of TLS in a way which authenticates only the passive entity of a TCPCL session or which does not authenticate either entity. Using TLS in a way which does not successfully authenticate some claim of both peer entities of a TCPCL session is outside of the scope of this document but does have similar properties to the opportunistic security model of [RFC7435].

8.12.2. Non-Certificate TLS Use

In environments where PKI is unavailable, alternate uses of TLS which do not require certificates such as pre-shared key (PSK) authentication [RFC5489] and the use of raw public keys [RFC7250] are available and can be used to ensure confidentiality within TCPCL. Using non-PKI node authentication methods is outside of the scope of this document.

8.13. Predictability of Transfer IDs

The only requirement on Transfer IDs is that they be unique with each session from the sending peer only. The trivial algorithm of the first transfer starting at zero and later transfers incrementing by one causes absolutely predictable Transfer IDs. Even when a TCPCL session is not TLS secured and there is a on-path attacker causing denial of service with XFER_REFUSE messages, it is not possible to preemptively refuse a transfer so there is no benefit in having unpredictable Transfer IDs within a session.

9. IANA Considerations

Registration procedures referred to in this section are defined in [RFC8126].

Some of the registries have been defined as version specific to TCPCLv4, and imports some or all codepoints from TCPCLv3. This was done to disambiguate the use of these codepoints between TCPCLv3 and TCPCLv4 while preserving the semantics of some of the codepoints.

9.1. Port Number

Within the port registry of [IANA-PORTS], TCP port number 4556 has been previously assigned as the default port for the TCP convergence layer in [RFC7242]. This assignment is unchanged by TCPCL version 4, but the assignment reference is updated to this specification. Each TCPCL entity identifies its TCPCL protocol version in its initial contact (see Section 9.2), so there is no ambiguity about what protocol is being used. The related assignments for UDP and DCCP port 4556 (both registered by [RFC7122]) are unchanged.

Parameter	Value
Service Name:	dtn-bundle
Transport Protocol(s):	TCP
Assignee:	IESG <iesg@ietf.org>
Contact:	IESG <iesg@ietf.org>
Description:	DTN Bundle TCP CL Protocol
Reference:	This specification.
Port Number:	4556

Table 10

9.2. Protocol Versions

IANA has created, under the "Bundle Protocol" registry [IANA-BUNDLE], a sub-registry titled "Bundle Protocol TCP Convergence-Layer Version Numbers". The version number table is updated to include this specification. The registration procedure is RFC Required.

Value	Description	Reference
0	Reserved	[RFC7242]
1	Reserved	[RFC7242]
2	Reserved	[RFC7242]
3	TCPCL	[RFC7242]
4	TCPCLv4	This specification.
5-255	Unassigned	

Table 11

9.3. Session Extension Types

EDITOR NOTE: sub-registry to-be-created upon publication of this specification.

IANA will create, under the "Bundle Protocol" registry [IANA-BUNDLE], a sub-registry titled "Bundle Protocol TCP Convergence-Layer Version 4 Session Extension Types" and initialize it with the contents of Table 12. The registration procedure is Expert Review within the lower range 0x0001--0x7FFF. Values in the range 0x8000--0xFFFF are reserved for use on private networks for functions not published to the IANA.

Specifications of new session extension types need to define the encoding of the Item Value data as well as any meaning or restriction on the number of or order of instances of the type within an extension item list. Specifications need to define how the extension functions when no instance of the new extension type is received during session negotiation.

Expert(s) are encouraged to be biased towards approving registrations unless they are abusive, frivolous, or actively harmful (not merely aesthetically displeasing, or architecturally dubious).

Code	Session Extension Type
0x0000	Reserved
0x0001--0x7FFF	Unassigned
0x8000--0xFFFF	Private/Experimental Use

Table 12: Session Extension Type Codes

9.4. Transfer Extension Types

EDITOR NOTE: sub-registry to-be-created upon publication of this specification.

IANA will create, under the "Bundle Protocol" registry [IANA-BUNDLE], a sub-registry titled "Bundle Protocol TCP Convergence-Layer Version 4 Transfer Extension Types" and initialize it with the contents of Table 13. The registration procedure is Expert Review within the lower range 0x0001--0x7FFF. Values in the range 0x8000--0xFFFF are reserved for use on private networks for functions not published to the IANA.

Specifications of new transfer extension types need to define the encoding of the Item Value data as well as any meaning or restriction on the number of or order of instances of the type within an extension item list. Specifications need to define how the extension functions when no instance of the new extension type is received in a transfer.

Expert(s) are encouraged to be biased towards approving registrations unless they are abusive, frivolous, or actively harmful (not merely aesthetically displeasing, or architecturally dubious).

Code	Transfer Extension Type
0x0000	Reserved
0x0001	Transfer Length Extension
0x0002--0x7FFF	Unassigned
0x8000--0xFFFF	Private/Experimental Use

Table 13: Transfer Extension Type Codes

9.5. Message Types

EDITOR NOTE: sub-registry to-be-created upon publication of this specification.

IANA will create, under the "Bundle Protocol" registry [IANA-BUNDLE], a sub-registry titled "Bundle Protocol TCP Convergence-Layer Version 4 Message Types" and initialize it with the contents of Table 14. The registration procedure is RFC Required within the lower range 0x01--0xEF. Values in the range 0xF0--0xFF are reserved for use on private networks for functions not published to the IANA.

Specifications of new message types need to define the encoding of the message data as well as the purpose and relationship of the new message to existing session/transfer state within the baseline message sequencing. The use of new message types need to be negotiated between TCPCL entities within a session (using the session extension mechanism) so that the receiving entity can properly decode all message types used in the session.

Expert(s) are encouraged to favor new session/transfer extension types over new message types. TCPCL messages are not self-delimiting, so care must be taken in introducing new message types.

If an entity receives an unknown message type the only thing that can be done is to send a MSG_REJECT and close the TCP connection; not even a clean termination can be done at that point.

Code	Message Type
0x00	Reserved
0x01	XFER_SEGMENT
0x02	XFER_ACK
0x03	XFER_REFUSE
0x04	KEEPALIVE
0x05	SESS_TERM
0x06	MSG_REJECT
0x07	SESS_INIT
0x08--0xEF	Unassigned
0xF0--0xFF	Private/Experimental Use

Table 14: Message Type Codes

9.6. XFER_REFUSE Reason Codes

EDITOR NOTE: sub-registry to-be-created upon publication of this specification.

IANA will create, under the "Bundle Protocol" registry [IANA-BUNDLE], a sub-registry titled "Bundle Protocol TCP Convergence-Layer Version 4 XFER_REFUSE Reason Codes" and initialize it with the contents of Table 15. The registration procedure is Specification Required within the lower range 0x00--0xEF. Values in the range 0xF0--0xFF are reserved for use on private networks for functions not published to the IANA.

Specifications of new XFER_REFUSE reason codes need to define the meaning of the reason and disambiguate it with pre-existing reasons. Each refusal reason needs to be usable by the receiving BP Agent to make retransmission or re-routing decisions.

Expert(s) are encouraged to be biased towards approving registrations unless they are abusive, frivolous, or actively harmful (not merely aesthetically displeasing, or architecturally dubious).

Code	Refusal Reason
0x00	Unknown
0x01	Completed
0x02	No Resources
0x03	Retransmit
0x04	Not Acceptable
0x05	Extension Failure
0x06	Session Terminating
0x07--0xEF	Unassigned
0xF0--0xFF	Private/Experimental Use

Table 15: XFER_REFUSE Reason Codes

9.7. SESS_TERM Reason Codes

EDITOR NOTE: sub-registry to-be-created upon publication of this specification.

IANA will create, under the "Bundle Protocol" registry [IANA-BUNDLE], a sub-registry titled "Bundle Protocol TCP Convergence-Layer Version 4 SESS_TERM Reason Codes" and initialize it with the contents of Table 16. The registration procedure is Specification Required within the lower range 0x00--0xEF. Values in the range 0xF0--0xFF are reserved for use on private networks for functions not published to the IANA.

Specifications of new SESS_TERM reason codes need to define the meaning of the reason and disambiguate it with pre-existing reasons. Each termination reason needs to be usable by the receiving BP Agent to make re-connection decisions.

Expert(s) are encouraged to be biased towards approving registrations unless they are abusive, frivolous, or actively harmful (not merely aesthetically displeasing, or architecturally dubious).

Code	Termination Reason
0x00	Unknown
0x01	Idle timeout
0x02	Version mismatch
0x03	Busy
0x04	Contact Failure
0x05	Resource Exhaustion
0x06--0xEF	Unassigned
0xF0--0xFF	Private/Experimental Use

Table 16: SESS_TERM Reason Codes

9.8. MSG_REJECT Reason Codes

EDITOR NOTE: sub-registry to-be-created upon publication of this specification.

IANA will create, under the "Bundle Protocol" registry [IANA-BUNDLE], a sub-registry titled "Bundle Protocol TCP Convergence-Layer Version 4 MSG_REJECT Reason Codes" and initialize it with the contents of Table 17. The registration procedure is Specification Required within the lower range 0x01--0xEF. Values in the range 0xF0--0xFF are reserved for use on private networks for functions not published to the IANA.

Specifications of new MSG_REJECT reason codes need to define the meaning of the reason and disambiguate it with pre-existing reasons. Each rejection reason needs to be usable by the receiving TCPCL Entity to make message sequencing and/or session termination decisions.

Expert(s) are encouraged to be biased towards approving registrations unless they are abusive, frivolous, or actively harmful (not merely aesthetically displeasing, or architecturally dubious).

Code	Rejection Reason
0x00	reserved
0x01	Message Type Unknown
0x02	Message Unsupported
0x03	Message Unexpected
0x04--0xEF	Unassigned
0xF0--0xFF	Private/Experimental Use

Table 17: MSG_REJECT Reason Codes

9.9. Object Identifier for PKIX Module Identifier

IANA has created, under the "Structure of Management Information (SMI) Numbers" registry [IANA-SMI], a sub-registry titled "SMI Security for PKIX Module Identifier". The table is updated to include a row "id-mod-dtn-tcpclv4-2021" for identifying the module in Appendix B as in the following table.

Decimal	Description	References
MOD-TBD	id-mod-dtn-tcpclv4-2021	This specification.

Table 18

9.10. Object Identifier for PKIX Other Name Forms

IANA has created, under the "Structure of Management Information (SMI) Numbers" registry [IANA-SMI], a sub-registry titled "SMI Security for PKIX Other Name Forms". The other name forms table is updated to include a row "id-on-bundleEID" for identifying DTN Endpoint IDs as in the following table.

Decimal	Description	References
ON-TBD	id-on-bundleEID	This specification.

Table 19

The formal structure of the associated other name form is in Appendix B. The use of this OID is defined in Section 4.4.1 and Section 4.4.2.

9.11. Object Identifier for PKIX Extended Key Usage

IANA has created, under the "Structure of Management Information (SMI) Numbers" registry [IANA-SMI], a sub-registry titled "SMI Security for PKIX Extended Key Purpose". The extended key purpose table is updated to include a purpose "id-kp-bundleSecurity" for identifying DTN endpoints as in the following table.

Decimal	Description	References
KP-TBD	id-kp-bundleSecurity	This specification.

Table 20

The formal definition of this EKU is in Appendix B. The use of this OID is defined in Section 4.4.2.

10. Acknowledgments

This specification is based on comments on implementation of [RFC7242] provided from Scott Burleigh.

11. References

11.1. Normative References

[IANA-BUNDLE]

IANA, "Bundle Protocol",
<<https://www.iana.org/assignments/bundle/>>.

[IANA-PORTS]

IANA, "Service Name and Transport Protocol Port Number Registry", <<https://www.iana.org/assignments/service-names-port-numbers/>>.

- [IANA-SMI] IANA, "Structure of Management Information (SMI) Numbers", <<https://www.iana.org/assignments/smi-numbers/>>.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<https://www.rfc-editor.org/info/rfc1122>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/info/rfc6066>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <<https://www.rfc-editor.org/info/rfc6960>>.

- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [I-D.ietf-dtn-bpbis]
Burleigh, S., Fall, K., and E. J. Birrane, "Bundle Protocol Version 7", Work in Progress, Internet-Draft, draft-ietf-dtn-bpbis-31, 25 January 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-dtn-bpbis-31>>.
- [X.680] ITU-T, "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2015, August 2015, <<https://www.itu.int/rec/T-REC-X.680-201508-I/en>>.

11.2. Informative References

- [AEAD-LIMITS]
Luykx, A. and K. Paterson, "Limits on Authenticated Encryption Use in TLS", August 2017, <<http://www.isg.rhul.ac.uk/~kp/TLS-AEbounds.pdf>>.
- [RFC2595] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, DOI 10.17487/RFC2595, June 1999, <<https://www.rfc-editor.org/info/rfc2595>>.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<https://www.rfc-editor.org/info/rfc3552>>.

- [RFC4511] Seremersheim, J., Ed., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, DOI 10.17487/RFC4511, June 2006, <<https://www.rfc-editor.org/info/rfc4511>>.
- [RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <<https://www.rfc-editor.org/info/rfc4838>>.
- [RFC5489] Badra, M. and I. Hajjeh, "ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)", RFC 5489, DOI 10.17487/RFC5489, March 2009, <<https://www.rfc-editor.org/info/rfc5489>>.
- [RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <<https://www.rfc-editor.org/info/rfc5912>>.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <<https://www.rfc-editor.org/info/rfc6698>>.
- [RFC7122] Kruse, H., Jero, S., and S. Ostermann, "Datagram Convergence Layers for the Delay- and Disruption-Tolerant Networking (DTN) Bundle Protocol and Licklider Transmission Protocol (LTP)", RFC 7122, DOI 10.17487/RFC7122, March 2014, <<https://www.rfc-editor.org/info/rfc7122>>.
- [RFC7242] Demmer, M., Ott, J., and S. Perreault, "Delay-Tolerant Networking TCP Convergence-Layer Protocol", RFC 7242, DOI 10.17487/RFC7242, June 2014, <<https://www.rfc-editor.org/info/rfc7242>>.
- [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <<https://www.rfc-editor.org/info/rfc7250>>.
- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <<https://www.rfc-editor.org/info/rfc7435>>.

- [RFC7457] Sheffer, Y., Holz, R., and P. Saint-Andre, "Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)", RFC 7457, DOI 10.17487/RFC7457, February 2015, <<https://www.rfc-editor.org/info/rfc7457>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8555] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J. Kasten, "Automatic Certificate Management Environment (ACME)", RFC 8555, DOI 10.17487/RFC8555, March 2019, <<https://www.rfc-editor.org/info/rfc8555>>.
- [I-D.ietf-dtn-bpsec]
III, E. J. B. and K. McKeever, "Bundle Protocol Security Specification", Work in Progress, Internet-Draft, draft-ietf-dtn-bpsec-27, 16 February 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-dtn-bpsec-27>>.
- [I-D.ietf-dtn-bibect]
Burleigh, S., "Bundle-in-Bundle Encapsulation", Work in Progress, Internet-Draft, draft-ietf-dtn-bibect-03, 18 February 2020, <<https://datatracker.ietf.org/doc/html/draft-ietf-dtn-bibect-03>>.
- [github-dtn-demo-agent]
Sipos, B., "TCPCL Example Implementation", <<https://github.com/BSipos-RKF/dtn-demo-agent/>>.
- [github-dtn-wireshark]
Sipos, B., "TCPCL Wireshark Dissector", <<https://github.com/BSipos-RKF/dtn-wireshark/>>.

Appendix A. Significant changes from RFC7242

The areas in which changes from [RFC7242] have been made to existing headers and messages are:

- * Split Contact Header into pre-TLS protocol negotiation and SESS_INIT parameter negotiation. The Contact Header is now fixed-length.
- * Changed Contact Header content to limit number of negotiated options.

- * Added session option to negotiate maximum segment size (per each direction).
- * Renamed "Endpoint ID" to "Node ID" to conform with BPv7 terminology.
- * Added session extension capability.
- * Added transfer extension capability. Moved transfer total length into an extension item.
- * Defined new IANA registries for message / type / reason codes to allow renaming some codes for clarity.
- * Segments of all new IANA registries are reserved for private/experimental use.
- * Expanded Message Header to octet-aligned fields instead of bit-packing.
- * Added a bundle transfer identification number to all bundle-related messages (XFER_SEGMENT, XFER_ACK, XFER_REFUSE).
- * Use flags in XFER_ACK to mirror flags from XFER_SEGMENT.
- * Removed all uses of SDNV fields and replaced with fixed-bit-length (network byte order) fields.
- * Renamed SHUTDOWN to SESS_TERM to deconflict term "shutdown" related to TCP connections.
- * Removed the notion of a re-connection delay parameter.

The areas in which extensions from [RFC7242] have been made as new messages and codes are:

- * Added contact negotiation failure SESS_TERM reason code.
- * Added MSG_REJECT message to indicate an unknown or unhandled message was received.
- * Added TLS connection security mechanism.
- * Added "Not Acceptable", "Extension Failure", and "Session Terminating" XFER_REFUSE reason codes.
- * Added "Resource Exhaustion" SESS_TERM reason code.

Appendix B. ASN.1 Module

The following ASN.1 module formally specifies the BundleEID structure, its Other Name form, and the bundleSecurity Extended Key Usage in the syntax of [X.680]. This specification uses the ASN.1 definitions from [RFC5912] with the 2002 ASN.1 notation used in that document.

```
<CODE BEGINS>
DTN-TCPCLv4-2021
  { iso(1) identified-organization(3) dod(6)
    internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-dtn-tcpclv4-2021(MOD-TBD) }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

IMPORTS
  OTHER-NAME
  FROM PKIX1Implicit-2009 -- [RFC5912]
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-implicit-02(59) }

  id-pkix
  FROM PKIX1Explicit-2009 -- [RFC5912]
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-explicit-02(51) } ;

id-kp OBJECT IDENTIFIER ::= { id-pkix 3 }

id-on OBJECT IDENTIFIER ::= { id-pkix 8 }

DTNOtherNames OTHER-NAME ::= { on-bundleEID, ... }

-- The otherName definition for Bundle EID
on-bundleEID OTHER-NAME ::= {
  BundleEID IDENTIFIED BY { id-on-bundleEID }
}

id-on-bundleEID OBJECT IDENTIFIER ::= { id-on ON-TBD }

-- Same encoding as GeneralName of uniformResourceIdentifier
BundleEID ::= IA5String

-- The Extended Key Usage key for bundle security
id-kp-bundleSecurity OBJECT IDENTIFIER ::= { id-kp KP-TBD }

END
<CODE ENDS>
```

Appendix C. Example of the BundleEID Other Name Form

EDITOR NOTE: The encoded hex part "0b" and OID segment "11" are to be replaced by ON-TBD allocated value. It was necessary to choose some OID value, so I chose the first not-allocated code point.

This non-normative example demonstrates an otherName with a name form of BundleEID to encode the Node ID "dtn://example/".

The hexadecimal form of the DER encoding of the otherName is:

a01c06082b0601050507080ba010160e64746e3a2f2f6578616d706c652f

And the text decoding in Figure 28 is an output of Peter Gutmann's "dumpasn1" program.

```
0 28: [0] {
2  8:  OBJECT IDENTIFIER '1 3 6 1 5 5 7 8 11'
12 16:  [0] {
14 14:    IA5String 'dtn://example/'
      :    }
      :  }
```

Figure 28: Visualized decoding of the on-bundleEID

Authors' Addresses

Brian Sipos
RKF Engineering Solutions, LLC
7500 Old Georgetown Road
Suite 1275
Bethesda, MD 20814-6198
United States of America

Email: brian.sipos+ietf@gmail.com

Michael Demmer
University of California, Berkeley
Computer Science Division
445 Soda Hall
Berkeley, CA 94720-1776
United States of America

Email: demmer@cs.berkeley.edu

Joerg Ott
Aalto University
Department of Communications and Networking
PO Box 13000
FI-02015 Aalto
Finland

Email: ott@in.tum.de

Simon Perreault
Quebec QC
Canada

Email: simon@per.reau.lt