NASA/TM-20240004702

High-Rate Delay Tolerant Networking (HDTN) Software Requirements Analysis

Rachel Dudukovich, Jose Lombay-Gonzalez, Stephanie Booth, Ethan Schweinsberg, Joseph Ponyik, Brian Tomko, and Nadia Kortas
Glenn Research Center, Cleveland, Ohio

Eric Brace
HX5, LLC, Brook Park, Ohio

Daniel Raible, Amber Waid, Alyssa Brewer, and Shira Nadile
Glenn Research Center, Cleveland, Ohio

May 2024

NASA STI Program Report Series

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION.
  Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM.
  Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT.
  Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONTRACTOR REPORT.
  Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION.
  Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION.
  Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION.
  English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

NASA/TM-20240004702

High-Rate Delay Tolerant Networking (HDTN) Software Requirements Analysis

Rachel Dudukovich, Jose Lombay-Gonzalez, Stephanie Booth, Ethan Schweinsberg, Joseph Ponyik, Brian Tomko, and Nadia Kortas
Glenn Research Center, Cleveland, Ohio

Eric Brace
HX5, LLC, Brook Park, Ohio

Daniel Raible, Amber Waid, Alyssa Brewer, and Shira Nadile
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

May 2024

Acknowledgments

The authors would like to thank the NASA Space Communications and Navigation (SCaN) program and the Delay Tolerant Networking (DTN) project for supporting this work, and in particular the continual guidance and management provided by John Nowakowski is most appreciated.

This report is available in electronic form at https://www.sti.nasa.gov/ and https://ntrs.nasa.gov/

NASA STI Program/Mail Stop 050
NASA Langley Research Center
Hampton, VA 23681-2199

CONTENTS

PREFACE

Space Communications and Navigation (SCaN) is developing new communications technologies to increase the amount of science data returned on future space missions. The High-Rate Delay Tolerant Networking (HDTN) project at NASA Glenn Research Center (GRC) will provide reliable internetworking as a high-speed path for moving data between spacecraft payloads, and across communication systems that operate on a range of different rates.

This document captures a subset of preliminary software requirements for the HDTN Computer Software Configuration Item (CSCI).

High-Rate Delay Tolerant Networking (HDTN) Software Requirements Analysis

Rachel Dudukovich, Jose Lombay-Gonzalez, Stephanie Booth, Ethan Schweinsberg, Joseph Ponyik, Brian Tomko, and Nadia Kortas
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Eric Brace
HX5, LLC
Brook Park, Ohio 44142

Daniel Raible, Amber Waid, Alyssa Brewer, and Shira Nadile
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

INTRODUCTION

This document serves as a detailed analysis of the main networking protocols implemented by HDTN. Sources of the protocol specifications include Internet Engineering Task Force (IETF) Request for Comments (RFC) and Consultative Committee for Space Data Systems (CCSDS) standards. The focus of this report is to derive software requirements suitable for NASA Procedural Requirements 7150.2D Class B compliance, including requirements traceability and software verification and validation, from the source specifications. This analysis will be incorporated into the finalized HDTN Software Requirements Specification (SRS) but does not encompass the full scope of the HDTN SRS. Requirements in this document are considered draft. The complete requirements will include bundle application requirements, interface requirements, computer resource requirements, software quality factors, and additional requirements as determined by the project. This document is publicly released to the greater community to receive feedback and foster collaboration opportunities.

Scope

Candidate requirements outlined in this document include bundle layer requirements, security requirements, convergence layer requirements and routing requirements. Formalized requirement levels are not captured in this document. Service level and network level requirements are out of the scope of this document.

Terminology

Requirement statements have been written using the word “shall” to indicate a planned process that must be implemented and verified. Original source documents include terms “must”, “must not”, “should”, “should not”, “may”, “required”, “shall”, and “recommended” as described in RFC 2119. In this document, “must”, “required”, and “shall” have been simplified to use only the term “shall”. Negative statements such as “shall not” have been rewritten as “shall”. These changes have been suggested per NASA software engineering best practices.

Methodology

Source RFC documents have been analyzed for all statements containing “shall”, “shall not”, “must”, “must not”, and “required”. These statements were considered candidate software requirements. Candidate software requirements were assessed according to the content of the statement and characterized as either “Requirement”, “Design Document”, “Data Dictionary”, “Requirement and Design Document”, or “Requirement and Data Dictionary”. Statements classified as “Requirement” describe functionality that must be implemented and verified. Statements classified as “Design Document” indicate information from the source documents that describe how a feature might be implemented, sections which describe procedures, or internal software mechanisms. Statements classified as “Data Dictionary” describe information regarding packet and field formats, initial value conditions, tables of values, and other information relevant to data types used in the implementation. Statements classified as “Data Dictionary” or “Design Document” information are out of the scope of this document but may be released in future documents. Requirements derived from RFCs were cross-referenced with requirements from the related CCSDS specifications when available.

REFERENCE DOCUMENTS

The following documents are referenced or provide guidance for this work.

NASA Process Documents

Document Number	Document Title	Release Date
NPR 7150.2D	*NASA Software Engineering Requirements*	*3/8/2022*
NASA-HDBK-2203B	*NASA Software Engineering and Assurance Handbook, Version B*	*4/20/2022*
HDTN-PLAN-004	*Software Configuration Management Plan*	*1/18/2024*
HDTN-PLAN-002	*Project Configuration Data Management Plan*	*11/8/2023*
HDTN-PLAN-006	*Software Assurance and Safety Plan*	*1/17/2024*
HDTN-PLAN-007	*HDTN NPR 7150.2D Compliance Matrix*	*5/2/2024*
GRC-SW-7150.12	*Formal Inspection and Peer Review*	

Protocol Specifications

Document Number	Document Title	Release Date
RFC 5050	*Bundle Protocol Version 6*	*11/2007*
CCSDS 734.2-B-1	*CCSDS Bundle Protocol Specification*	*9/2015*
RFC 9171	*Bundle Protocol Version 7*	*12/14/2022*
CCSDS 734.2-P-1.1	*CCSDS Bundle Protocol Specification*	*4/2023*
RFC 9172	*Bundle Protocol Security (BPSec)*	*10/11/2023*
RFC 9173	*Default Security Contexts for Bundle Protocol Security (BPSec)*	*6/21/2022*
CCSDS 734.5-R-2	*CCSDS Bundle Protocol Security Specification*	*9/2023*
RFC 5326	*Licklider Transmission Protocol*	*12/8/2022*
CCSDS 734.1-B-1	*Licklider Transmission Protocol (LTP) for CCSDS*	*5/2015*
RFC 9174	*DTN TCP Convergence-Layer Protocol Version 4*	*1/31/2022*
RFC 7122	*Datagram Convergence Layers for DTN Bundle Protocol and Licklider Transmission Protocol*	*3/2014*
draft-burleigh-dtnrg-cgr-01	*Contact Graph Routing*	*7/8/2010*
CCSDS 734.3-B-1	*Schedule-Aware Bundle Routing*	*7/2019*
RFC 2119	*Key words for use in RFCs to Indicate Requirement Levels*	*3/1997*

1 BPV6 REQUIREMENTS

BPv6-001 URI Scheme Specific Part

The Scheme Specific Part of every URI formed within the "ipn" scheme shall comprise:

1. The node number of the URI.

2. An ASCII period ('.') character.

3. The service number of the URI.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	The scheme specific part is defined in RFC 6260 section 2.1. Following the same format between bundles will aide in consistency within the network.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from BPGen, BPSendFile, and HDTN and is shown to contain URI's that follow the specified "ipn" scheme.

BPv6-002 CBHE Unit

A compressed primary block shall contain integers for its fields.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	The CBHE-compressed primary block is defined in RFC 6260 section 2.2. If the value is at the null endpoint then that piece of the primary block is zero.
Verification Method:	Inspection
Verification Approach:	This requirement is considered verified when an inspection of the code shows that, when compressing a bundle, the primary block fields are integers.

BPv6-003 CBHE Encoding of EIDs by CLAs

A compressed primary block shall, in order, contain:

1. The node number of the destination endpoint ID.

2. The service number of the destination endpoint ID.

3. The node number of the source endpoint ID.

4. The service number of the source endpoint ID.

5. The node number of the report-to-endpoint ID.

6. The service number of the report-to-endpoint ID.

7. The node number of the current custodian endpoint ID.

8. The service number of the current custodian endpoint ID.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	The CBHE-compressed primary block is defined in RFC 6260 section 2.2. If the value is at the null endpoint then that piece of the primary block is zero.

Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from BPGen, BPSendFile, and HDTN and confirms the compressed primary block contains each field in the specified order.

BPv6-004 BP Time

DTN time shall consist of nanosecond precision since the start of the year 2000.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	BP Time is defined in CCSDS 734.2-B-1 section 3.4 as well as RFC 5050 section 6.1. Time must be in the proper format down to the nanoseconds. If the node's time system does not provide sufficient precision then the precision of the onboard system can be used.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from BPGen, BPSendFile, and HDTN and confirms DTN time within the bundle consists of the number of seconds since the start of the year 2000 down to the number of nanoseconds since the start of the indicated second.

BPv6-005 LTP Bundle Encapsulation

When utilizing LTP bundles, the bundle shall be entirely contained in a single LTP session of red-data.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	LTP transmission is defined in CCSDS 734.2-B-1 section B3.1.2.2. For reliable bundle transmission, bundles are encapsulated in LTP blocks containing only red-part (reliable) data in either of these two different encapsulations. Reliable transmission is a key component in a network. LTP blocks are organized according to the Client Operations section 7 of the LTP-for_CCSDS Book.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when LTP bundles are shown to be encapsulated as a LTP single bundle block without leading and trailing bytes when the bundle is to be a single LTP bundle.

BPv6-006 Single LTP Bundle Encapsulation

When single bundle per LTP block is selected, bundles shall be encapsulated as a single bundle.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	LTP CL adaptor is detailed in section B3.1.2.2 in CCSDS 734.2-B-1. An LTP bundle block utilizes the Destination LTP Client Service ID for ‘Bundle Protocol’ as specified in the SANA LTP Client Service ID Number Registry (reference [5]).
Verification Method:	Test
Verification Approach:	This requirement is considered verified when the LTP block is encapsulated as a single bundle when the bundle is to be a single LTP bundle.

BPv6-007 UDP CL Adaptor

When using the convergence layer, UDP, encapsulated bundles shall be in UDP datagrams.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	UDP convergence layer adapter is defined in CCSDS 734.2-B-1 section B4; ensures network cohesion.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from BPGen, BPSendFile, and HDTN via the UDP convergence layer and confirms the bundle contains a single UDP datagram with a checksum and the specified UDP port is utilized.

BPv6-008 BP Initialization Registration Service

The Bundle Protocol services shall initiate a registration (registering a node in an endpoint) when requested.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	Services at the user interface is defined in CCSDS 734.2-B-1 section 4.1.1. Bringing the capability to the user is imperative for adding and subtracting data to and from the network.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when users can create a registration.

BPv6-009 BP Terminate Registration Service

The Bundle Protocol services shall initiate termination of a registration when requested.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	Services at the user interface is defined in CCSDS 734.2-B-1 section 4.1.1. Bringing the capability to the user is imperative for adding and subtracting data to and from the network.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when users can terminate a registration.

BPv6-010 BP Transmit Bundle Service

The Bundle Protocol services shall transmit a bundle to an identified bundle endpoint when requested.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	Services at the user interface is defined in CCSDS 734.2-B-1 section 4.1.1. Bringing the capability to the user is imperative for adding and subtracting data to and from the network.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when users can transmit a bundle.

BPv6-011 BP Cancel Transmission Service

The Bundle Protocol services shall initiate cancelation of a transmission when requested.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	Services at the user interface is defined in CCSDS 734.2-B-1 section 4.1.1. Bringing the capability to the user is imperative for adding and subtracting data to and from the network.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when users can cancel a transmission.

BPv6-012 BP Deliver Received Bundle Service

The Bundle Protocol services shall initiate delivery of a received bundle when requested.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	Services at the user interface is defined in CCSDS 734.2-B-1 section 4.1.1. Bringing the capability to the user is imperative for adding and subtracting data to and from the network.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when users can initiate delivery of a received bundle.

BPv6-013 BP Bundle Structure

Bundles shall utilize SDNV encoding.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	To understand the header and data within the network, encoding needs to be decided upon. SDNV is the encoding for bundle fields for BP v6.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from BPGen, BPSendFile, and HDTN and confirms the bundle is encapsulated with valid SDNV encoding.

BPv6-014 BP Bundle Block Structure

Bundles shall be a concatenated sequence of two or more block structures.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 4
Rationale:	Every bundle needs at least a payload block and a primary block per RFC 5050. A bundle can have additional block per requests.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when an initiated bundle contains two or more concatenated block structures.

BPv6-015 Primary Bundle Location

The first block in the bundle shall be the primary bundle block.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 4
Rationale:	The location of the primary bundle block is defined in RFC 5050 section 4.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when an initiated bundle contains a primary bundle block in the first block of the bundle.

BPv6-016 Number of Primary Blocks

A bundle shall have one primary bundle block.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 4
Rationale:	Number of primary bundle blocks is defined in RFC 5050 section 4. All main and required header information is placed in the primary bundle block. Therefore, having more would be redundant.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a bundle is analyzed to have only one primary bundle block.

BPv6-017 Last Block Flag

The last block in the sequence shall have the "last block" flag (in its block processing control flags) set to 1.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 4
Rationale:	The "last block" flag is defined in RFC 5050 section 4. Setting this flag shows that the data stream is ending at the block. Therefore, this flag shall be set to zero for every block in the bundle after the primary block except the final block, which must be set to zero.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a bundle is analyzed with the last block in the sequence to have the "last block" flags set to 1.

BPv6-018 Custody Transfer Request Flag of an Administrative Record

The custody transfer requested flag shall be zero when the bundle is an administrative record.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 4.2
Rationale:	Bundle processing control flags are defined in RFC 5050 section 4.2. Network management is necessary for determining the health and statistics of the network.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives an administrative record bundle from HDTN and confirms the bundle has the custody transfer requested flag set to zero.

BPv6-019 Status Report Request Flag of an Administrative Record

Status report request flags shall be zero when the bundle is an administrative record.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 4.2
Rationale:	Bundle processing control flags are defined in RFC 5050 section 4.2. Network management is necessary for determining the health and statistics of the network.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a received administrative record bundle has all status report request flags set to zero.

BPv6-020 Disable Identity of Bundle Protocol Features

When the bundle's source endpoint ID is "dtn:none", then bundle identity shall be disabled by setting its bundle protocol features:

1.The bundle's custody transfer requested flag is set to zero,

2.The "Bundle must not be fragmented" flag is set to 1, and

3.Remaining status report request flags must be zero.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 4.2
Rationale:	Bundle processing control flags are defined in RFC 5050 section 4.2. A "dtn:none" source endpoint ID means the bundle is not uniquely identifiable. The features that rely on bundle identity are the bundle's custody transfer requested flag must be zero, the "Bundle must not be fragmented" flag must be 1, and all status report request flags must be zero.
Verification Method:	Inspection
Verification Approach:	The requirement is considered verified when a code inspection shows a bundle's identity is disabled when the endpoint ID="din:none", the custody transfer flag = 0; the "Bundle must not be fragmented" flag = 1, and remaining status report flags = 0.

BPv6-021 Primary Bundle Block

Bundles created by a source bundle protocol agent shall have a unique combination of source endpoint ID and bundle creation timestamp.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 4.5.1
Rationale:	Creation timestamp is defined in RFC 5050 section 4.5.1. The combination of source endpoint ID and bundle creation timestamp therefore serves to identify a single transmission request, enabling it to be acknowledged by the receiving application (provided the source endpoint ID is not "dtn:none").
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives multiple bundles from HDTN and confirms all bundles have a unique combination of source endpoint ID and bundle creation timestamp.

BPv6-022 Constraining Bundle Fragmentation 1

The concatenation of fragments shall result in a payload identical to the bundle that was fragmented.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.8
Rationale:	Bundle fragmentation is defined in RFC 5050 section 5.8. The payloads of fragments resulting from different fragmentation episodes, in different parts of the network, may be overlapping subsets of the original bundle's payload.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a received reassembled bundle's payload is identical to the original payload prior to fragmentation.

BPv6-023 Constraining Bundle Fragmentation 2

The primary block's bundle processing flags of each fragment shall indicate that the bundle is a fragment.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.8
Rationale:	Bundle structure is defined in RFC 5050 section 5.8. Keeping track of fragmented bundles is imperative for network management.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a fragmented bundle from BPGen, BPSendFile, and HDTN and confirms the bundle has the primary block bundle processing flags set to true.

BPv6-024 Constraining Bundle Fragmentation 3

The end of each fragment's primary bundle block shall contain the fragment offset followed by the total application data unit length.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.8
Rationale:	Bundle structure is defined in RFC 5050 section 5.8. Keeping track of fragmented bundles is imperative for network management. The fragment offset and total application data unit length aids in reconstructing the bundle once it reaches its destination.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a fragmented bundle from BPGen, BPSendFile, and HDTN and confirms the end of each received fragment contains the fragment offset followed by the total application data unit length.

BPv6-025 Bundle Fragmentation Replication

When the 'Block must be replicated in every fragment' bit is 1, then the block shall be replicated in the fragment(s).

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.8
Rationale:	Bundle structure is defined in RFC 5050 section 5.8. The "Block must be replicated in every fragment" bit, when true, means that the block shall be replicated in every fragment.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a fragmented bundle from BPGen, BPSendFile, and HDTN and confirms the bundle contains a block with the specified bit set to 1 and the specified block is shown to be replicated in each fragment.

BPv6-026 Bundle Fragmentation Order

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

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.8
Rationale:	Bundle structure is defined in RFC 5050 section 5.8. Keeping block order during fragmentation is imperative for network management; will aid in reconstructing the bundle once it reaches its destination.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when fragmenting a bundle shows the blocks are kept in the same order as the bundle prior to fragmentation.

BPv6-027 SDNV MSB

An SDNV encoded in N octets shall have the MSB of every octet set to 1, excluding the last octet.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 4.1
Rationale:	SDNVs are defined in RFC 5050 section 4.1. An SDNV is a numeric value. The value encoded in an SDNV is the unsigned binary number obtained by concatenating into a single bit string the 7 least significant bits of each octet of the SDNV. The last of an SDNV string has its most significant bit (MSB) set to zero.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a received bundle is encoded in N octets with the MSB of every octet set to 1 excluding the last octet.

BPv6-028 Dictionary Revision

When removal of a string from the dictionary changes the offsets of other strings in the dictionary, endpoint ID references that refer to the other strings shall be adjusted.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 4.7
Rationale:	Dictionary revision(s) are defined in RFC 5050 section 4.7. Note that these references may be in the primary block and/or in the block EID reference fields of extension blocks.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when the removal of a string from the dictionary that causes the offsets of other strings to change results in endpoint ID references that refer to those strings being adjusted.

BPv6-029 Bundle Reception Unintelligible

The bundle protocol agent shall delete the bundle for the reason "Block unintelligible" for any extension block that the bundle protocol agent cannot process.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.6
Rationale:	To keep the network efficient, anything the network doesn't recognize will be discarded.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends a bundle with a corrupt extension block to HDTN and is confirmed that block has been deleted.

BPv6-030 Removal of a Block from a Bundle

When the block is indicated to be discarded then the bundle protocol agent shall remove this block from the bundle.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.6
Rationale:	To keep the network efficient, anything the network doesn't recognize will discard. This indication is shown by the block processing flags. The criteria to remove the block from the bundle is if the 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.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends a bundle containing an "un-processable" block with the specified flag set to HDTN and the block is shown to have been deleted.

BPv6-031 Bundle Reception Block Process Flag

The bundle protocol agent shall set the "Block was forwarded without being processed" flag to 1 in the block processing flags of the block when the block cannot be processed.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.6
Rationale:	The "block was forwarded without being processed" flag will not be 1 if the block processing flags in that block indicate that the bundle must be deleted or that the block must be discarded.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a bundle with an un-processable block is sent to HDTN and the block is shown to have been updated with the "Block was forwarded without being processed" flag set to 1.

BPv6-032 Bundle Reception Custody Failure

For a singleton endpoint bundle with custody transfer redundancy, the bundle protocol agent shall generate a "Failed" custody signal via a flag.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.6
Rationale:	Procedures for handling redundancy in custody transfer for a bundle whose destination is not a singleton endpoint are not defined in the RFC 5050 specification. The specific flag to be set for this bundle is "redundant reception" reason code. This happens when receiving a redundant single endpoint bundle.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends a singleton endpoint bundle with custody transfer redundancy to HDTN and confirms a "Failed" custody signal is generated with the "redundant reception" flag set.

BPv6-033 Reporting of Custodial Delivery

When the bundle's custody transfer requested flag (in the bundle processing flags field) is set to 1, custodial delivery shall be reported.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.7
Rationale:	Procedures for reporting custodial delivery for a bundle whose destination is not a singleton endpoint are not defined in the RFC 5050 specification. Network management is necessary for determining the health and statistics of the network.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends a bundle to HDTN with the "custody transfer requested" flag set and confirms that custodial delivery is reported.

BPv6-034 Local Bundle Delivery Succeeded Response

When a bundle contains a singleton endpoint destination, the bundle protocol agent shall report custodial delivery by generating a "Succeeded" custody signal for the bundle, destined for the bundle's current custodian.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.7
Rationale:	When custody is enabled a handshake happens between the two nodes. Details of bundle acceptance is defined in RFC 5050 section 5.7.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends a single destination bundle with custody transfer enabled to HDTN and confirms a "succeeded" custody signal is generated.

BPv6-035 Block Processing Control Flags Zeroing Flags

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

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 4.3
Rationale:	When a bundle is fragmented, it fragments itself at the payload block therefore fragmenting all blocks after the payload block too. Bundle transmission service is defined in RFC 5050 section 4.3.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a fragmented bundle from BPGen, BPSendFile, and HDTN and confirms all fragments have the specified bit set to zero for blocks following the payload block.

BPv6-036 Bundle Transmission Commit to Custody

When custody transfer is requested, the bundle protocol agent shall commit to accepting custody of the bundle.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.2
Rationale:	When custody is enabled, custody rules remain through the lifetime of the bundle. In terms of bundle transmission services regarding custody transfer are defined in RFC 5050 section 5.2.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends a bundle with custody to HDTN and confirms that HDTN commits to accepting custody of the bundle.

BPv6-037 Bundle Transmission Source Endpoint

The source endpoint ID of the bundle shall be either the ID of an endpoint of which the node is a member or the null endpoint ID "dtn:none".

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.2
Rationale:	The bundle transmission service details are defined in RFC 5050 section 5.2. The source endpoint ID is the same ID as the node that created it. This allows network users to pinpoint which node created the bundle.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives of bundle with custody from BPGen, BPSendFile, and HDTN and confirms the bundle has a source endpoint ID of the sending node.

BPv6-038 Bundle Forwarding Contraindication Lookup

When the bundle's Status Report Reason Codes are zero, the bundle protocol agent shall allow a bundle to be forwarded.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.4
Rationale:	Contraindication reasons are defined in Figure 12 of RFC 5050. Contraindication is placed upon a bundle whenever the bundle protocol agent needs to do something else with the bundle for any reason.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends a bundle with status report reason codes set to zero to HDTN and confirms the bundle is forwarded.

BPv6-039 Bundle Forwarding Choosing Endpoint

The bundle protocol agent shall determine the endpoint(s) to forward the bundle to.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.4
Rationale:	The bundle protocol agent may choose either to forward the bundle directly to its destination endpoint (if possible) or to forward the bundle to some other endpoint(s) for further forwarding. The manner in which this decision is made may depend on the scheme name in the destination endpoint ID but is beyond the scope of this document. If the agent finds it impossible to select any endpoint(s) to forward the bundle to, then forwarding is contraindicated.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends a bundle to HDTN and confirms HDTN sets the next hop of the bundle.

BPv6-040 Bundle Forwarding Block Sequencing

The sequencing of the blocks in a forwarded bundle shall remain the same as the bundle transits a node.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.4
Rationale:	Bundle forwarding block sequencing is defined in RFC 5050 section 5.4. Keeping the block sequencing between hops will avoid the possibility of invalidating bundle security.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends a bundle to HDTN and confirms HDTN does not alter the order of the bundle blocks.

BPv6-041 Bundle Forwarding Reason Code

When the bundle requires a reason code without any constraints, then the reason code shall be "no additional information".

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.4
Rationale:	Bundle forwarding is defined in RFC 5050 section 5.4. Bundles may require reason codes even if it is not contraindicated.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle status report from BPGen, BPSendFile, and HDTN and confirms the reason code is set to "no additional information" by default.

BPv6-042 Forwarding Contraindicated Decision

The bundle protocol agent shall determine when forwarding is contraindicated based on RFC 5050 Figure 12: Status Report Reason Codes.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.4.1
Rationale:	Bundle forwarding is defined in RFC 5050 section 5.4.1. If the bundle protocol agent decides that the bundle failed to forward, it is likely to be influenced by the reason for which forwarding is contraindicated.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when the failure signal is received and then the bundle is resent.

BPv6-043 Forwarding Failed Custody Signal Current Custodian

When a bundle has a singleton endpoint destination, the bundle protocol agent shall handle the custody transfer failure by generating a "Failed" custody signal for the bundle that is destined for the bundle's current custodian.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.4.2
Rationale:	When custody is enabled, custody rules remain through the lifetime of the bundle. Custody deletion is defined in RFC 5050 section 5.4.2.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends a bundle with custody to an HDTN node that is unable to accept custody and confirms that a "failed" custody signal is generated.

BPv6-044 Forwarding Failed Reason Code

When forwarding fails, the custody signal shall contain a reason code indicating the reason for the failure.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.4.2
Rationale:	Bundle forwarding rules are defined in RFC 5050 section 5.4.2. The custody signal contains, when applicable, the contraindication when a bundle uses reliability.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends a bundle with custody to an HDTN node that is unable to forward it and confirms a failed custody signal is generated containing the reason code for the failure.

BPv6-045 Bundle Expiration

When a bundle expires, the bundle protocol agent shall delete the bundle for the reason "lifetime expired".

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.5
Rationale:	Bundle forwarding rules for expired bundles are defined in RFC 5050 section 5.5. Network needs not to waste resources moving expired data.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends an expired bundle to HDTN and confirms the expired bundle is deleted from HDTN storage.

BPv6-046 Bundle Discard Prevented

When deletion of the singleton endpoint bundle is prevented from being discarded, then a bundle deletion status report citing the reason for deletion shall be generated.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.13
Rationale:	Bundle deletion is defined in RFC 5050 section 5.13. There are extra procedures for bundles which utilize reliability.
Verification Method:	Inspection
Verification Approach:	This requirement is considered verified when a code inspection shows a bundle deletion status report when a bundle is prevented from being discarded.

BPv6-047 Bundle Deletion

To delete a bundle, a bundle's retention constraints shall be removed.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.13
Rationale:	Bundle deletion procedures are defined in RFC 5050 section 5.13. Network management needs to account for data delivered.
Verification Method:	Inspection
Verification Approach:	This requirement is considered verified when a code inspection shows the removal of a bundle's retention constraints.

BPv6-048 Generation of Bundle Deletion Status Report

A bundle deletion status report shall be generated after the deletion of a bundle for which custody transfer is requested.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	Administrative record definitions are defined in RFC 5050. Reports aid in network statistics.
Verification Method:	Inspection
Verification Approach:	This requirement is considered verified when a code inspection shows a generation of a bundle deletion status report when custody transfer was requested.

BPv6-049 Intermittent Connectivity Conditions

The DTN implementation shall store data when there is intermittent connectivity.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 1
Rationale:	DTN is store, carry, forward. This is due to the intermittent nature that DTN is used.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends a bundle to an HDTN node with no other connectively and confirms that the bundle is stored.

BPv6-050 Late Binding

The DTN implementation shall provide dynamic EID binding capabilities.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 1
Rationale:	Late binding capabilities are mentioned in RFC 5050 section 1.
Verification Method:	Inspection
Verification Approach:	This requirement is considered verified when a code inspection shows a dynamic EID binding capability.

BPv6-051 Custody Acceptance Succeeded Signal

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

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6, RFC 5050 Section 5.10.1
Rationale:	When custody is enabled, a handshake happens between the two nodes. Details of custody signal processing is defined in RFC 5050 section 5.10.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends a bundle with custody to HDTN and confirms a "succeeded" custody signal is generated.

BPv6-052 Retransmit After Custody Transfer Timer Expiration

When custody transfer has failed by the custody transfer timer expiration, the DTN implementation shall retransmit the bundle.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	When custody is enabled, a handshake happens between the two nodes.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a bundle with custody is retransmitted when the bundle had an expired custody transfer timer expiration.

BPv6-053 Retransmit After Custody Signal Reception Failure

When a failed custody signal has been received, the bundle agent shall retransmit the bundle.

Status:	Vetted
Source:	CCSDS 734.2-B-1 Table A6
Rationale:	When custody is enabled, a handshake happens between the two nodes.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a bundle with custody is retransmitted.

BPv6-054 Bundles Retained for Forwarding

Bundle state information of BP nodes shall provide the number of bundles in storage at the current node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F1 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided from the current node providing the number of bundles in storage.

BPv6-055 Bundles Retained for Transmission

Bundle state information of BP nodes shall provide the number of bundles queued for transmission at the current node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F1 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided from the current node providing the number of bundles queued for transmission.

BPv6-056 Bundles Retained for Custody Acceptance

Bundle state information of BP nodes shall provide the number of bundles retained for custody acceptance at the current node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F1 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided from the current node providing the number of bundles retained for custody acceptance.

BPv6-057 Bundles Retained for Reassembly

Bundle state information of BP nodes shall provide the number of bundles retained for reassembly at the current node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F1 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided from the current node providing the number of bundles retained for reassembly.

BPv6-058 Bulk Bundles Sourced

Bundle state information of BP nodes shall provide the number of bundles generated by the current node with the bulk priority.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F1 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided from the current node providing the number of bundles generated with bulk priority.

BPv6-059 Normal Bundles Sourced

Bundle state information of BP nodes shall provide the number of bundles generated by the current node with the normal priority.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F1 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided from the current node providing the number of bundles generated with normal priority.

BPv6-060 Expedited Bundles Sourced

Bundle state information of BP nodes shall provide the number of bundles generated by the current node with the expedited priority.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F1 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided from the current node providing the number of bundles generated with expedited priority.

BPv6-061 Bulk Bundles Queued

Bundle state information of BP nodes shall provide the number of bundles with the bulk priority currently resident on the node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F1 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided from the current node providing the number of bundles with bulk priority.

BPv6-062 Normal Bundles Queued

Bundle state information of BP nodes shall provide the number of bundles with the normal priority currently resident on the node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F1 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided from the current node providing the number of bundles with normal priority.

BPv6-063 Expedited Bundles Queued

Bundle state information of BP nodes shall provide the number of bundles with the expedited priority currently resident on the node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F1 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided from the current node providing the number of bundles with expedited priority.

BPv6-064 Fragmentation

Bundle state information of BP nodes shall provide the number of bundles that have been fragmented by this node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F1 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided with the number of bundles that have been fragmented by the tested node.

BPv6-065 Number of Fragments

Bundle state information of BP nodes shall provide the number of fragments created by this bundle.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F1 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided with the number of fragments which were created by the tested node.

BPv6-066 Failed Custody Transfers

Error and reporting information of BP nodes shall provide the number of custody failed incoming bundles at this node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F2 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided with the number of incoming bundles that had a request for custody failed at the tested node.

BPv6-067 Failed Forwards

Error and reporting information of BP nodes shall provide the number of bundles that have experienced a forwarding failure at this node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F2 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided with the number of incoming bundles that had experienced a forwarding failure at the node.

BPv6-068 Abandoned Delivery

Error and reporting information of BP nodes shall provide the number of abandoned bundles at this node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F2 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided with the number of incoming bundles that had delivery abandoned at the node.

BPv6-069 Discarded Bundles

Error and reporting information of BP nodes shall provide the number of bundles discarded at this node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F2 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided with the number of bundles that have been discarded at the node.

BPv6-070 Activity State

Registration information of BP nodes shall provide the Endpoint ID of this registered endpoint.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F3 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided containing the endpoint ID of the registered endpoint.

BPv6-071 Singleton State

Registration information of BP nodes shall provide the current state of the EID, at the time the managed information was queried.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F3 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided with the queried current state of the EID.

BPv6-072 Default Failure Action

Registration information of BP nodes shall provide whether this EID is a singleton EID.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F3 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided with the node's EID classification.

BPv6-073 Node Identifier

Registration information of BP nodes shall provide the default action to be taken when delivery is not possible.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F3 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided with the default action the node takes when delivery is not possible.

BPv6-074 Bundle Protocol Version Number

Node state information of BP nodes shall provide the Endpoint ID that uniquely and permanently identifies this node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F4 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided with the node's Endpoint ID that is unique and permanent to the node.

BPv6-075 Available Storage

Node state information of BP nodes shall provide the number of the version of the Bundle Protocol that is supported at this node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F4 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided with the supported BP version for the node under test.

BPv6-076 Last Up Time 1

Node state information of BP nodes shall provide the number of kilobytes of storage allocated to bundle retention at this node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F4 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report provides the number of kilobytes of storage allocated to bundle retention and the number of kilobytes of storage available to store bundles at the node under test.

BPv6-077 Last Up Time 2

Node state information of BP nodes shall provide the number of kilobytes of storage available to store bundles at this node and the number of kilobytes of storage available to store bundles.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F4 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report provides the number of kilobytes of storage allocated to bundle retention and the number of kilobytes of storage available to store bundles at the node under test.

BPv6-078 Extension Name

Node state information of BP nodes shall provide the name identifying one of the BP extensions supported per extension at this node.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and MIB
Rationale:	This requirement is stated in the CCSDS blue book table F4 for standardized metrics. Utilizing standardized metrics aids in network management and interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when a report is provided with the node's BP supported extensions.

BPv6-079 State of the Node Managed Information

BP node(s) shall support a set of managed information that represents the state of the node at a particular time.

Status:	Draft
Source:	CCSDS 734.2-B-1 Table A6 and A6
Rationale:	Adding time into the managed information aids in statistical analysis and/or network health monitoring.
Verification Method:	Inspection
Verification Approach:	This requirement is considered verified when a code inspection shows a set of managed information that is to be gathered.

2 BPV7 REQUIREMENTS

BPv7-001 CBOR Encoding

The data type for bundle block fields shall be CBOR unsigned integers.

Status:	Vetted
Source:	RFC 9171 Section 4
Rationale:	CBOR is the encoding for bundle fields. BP v7 only uses unsigned integers, byte strings, and array.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when an inspection of the code shows that data type for bundle block fields are CBOR unsigned integers.

BPv7-002 Bundle Structure

Bundles shall be a CBOR indefinite-length array.

Status:	Vetted
Source:	RFC 9171 Section 4.1
Rationale:	CBOR is the encoding for bundle fields. BP v7 only uses unsigned integers, byte strings, and array. Bundle structure is defined in RFC 9171 section 4.1.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when an inspection of the code shows that bundles are a CBOR indefinite-length array.

BPv7-003 Bundle Array Structure

Bundle arrays shall have a length of at least two blocks.

Status:	Vetted
Source:	RFC 9171 Section 4.1
Rationale:	Primary bundle block and payload bundle block is the minimum requirement for a bundle. Bundle structure is defined in RFC 9171 section 4.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from HDTN and confirms that the bundle array has a length of at least two blocks.

BPv7-004 Block Order

Block(s) following the primary block shall be a canonical block.

Status:	Vetted
Source:	RFC 9171 Section 4.1
Rationale:	Canonical blocks are the extension blocks for when data is added to the bundle header total for extra information. Block order is defined in RFC 9171 section 4.1 and 4.3.2.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from HDTN and confirms the block following the primary block conforms to the canonical block format.

BPv7-005 Bundle Structure Stop Code

A CBOR "break" stop code, terminating the array, shall be directly after the payload block.

Status:	Vetted
Source:	RFC 9171 Section 4.1
Rationale:	The stop code tells the node that the bundle is finished. Block order is defined in RFC 9171 section 4.1
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from HDTN and confirms the payload block is terminated by the CBOR "break" stop code.

BPv7-006 Bundle Encoding

The CBOR encodings values of block fields shall conform to the core deterministic encoding requirements specified in the data dictionary, except indefinite-length items.

Status:	Vetted
Source:	RFC 9171 Section 4.1
Rationale:	Indefinite length items are not prohibited within the BPv7 specification. CBOR, from RFC 8949, is specified as the encoding scheme per RFC 9171.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when an inspection of the code shows all values in block fields are CBOR encodings.

BPv7-007 Primary Bundle Location

The first block in the bundle shall be a primary bundle block.

Status:	Vetted
Source:	RFC 9171 Section 4.1
Rationale:	The location of the primary bundle block is defined in RFC 9171 section 4.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from HDTN a confirms the first block is a primary bundle block.

BPv7-008 Number of Primary Blocks

A bundle shall have exactly one primary bundle block.

Status:	Vetted
Source:	RFC 9171 Section 4.1
Rationale:	Number of primary bundle blocks is defined in RFC 9171 section 4.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from HDTN and confirms there is only one primary bundle block.

BPv7-009 Payload Block Location

The last bundle block shall be a payload block.

Status:	Vetted
Source:	RFC 9171 Section 4.1
Rationale:	The location of the payload block is defined in RFC 9171 section 4.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from HDTN and confirms the last block in the bundle is a payload block.

BPv7-010 Number of Payload Blocks

A bundle shall have exactly one payload block.

Status:	Vetted
Source:	RFC 9171 Section 4.1
Rationale:	The payload contains the message which can be either data, information, or communication. Number of payload bundle blocks is defined in RFC 9171 section 4.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from HDTN and confirms the bundle has exactly one primary bundle block.

BPv7-011 CRC Types

CRC type will specify omitted, X-25 CRC-16, or CRC32C CRC-32.

Status:	Vetted
Source:	RFC 9171 Section 4.2.1
Rationale:	CRC type is defined in RFC 9171 section 4.2 and 4.3.1. CRC values explain if CRC is calculated or not.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when an inspection of the code shows the CRC type as omitted, X-25 CRC-16, or CRC32C CRC-32.

BPv7-012 Bundle Processing Control Flags

The bundle processing control flags shall be processed as a bit field.

Status:	Draft
Source:	RFC 9171 Section 4.2.3
Rationale:	Bit field values are defined in the data dictionary. Control flag values are defined in RFC 9171 section 4.2.3 with details on page 14.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when an inspection of the code shows a bit field for the bundle processing control flags.

BPv7-013 Control Flags for ADU

When bundle processing control flag "ADU is an administrative record" bit is TRUE, then all status report request flag values shall be FALSE.

Status:	Draft
Source:	RFC 9171 Section 4.2.3
Rationale:	Control flag values are defined in RFC 9171 Section 4.2.3.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle with the "ADU is administrative record" bit set to TRUE and confirms that all status report request flags values are set to 0.

BPv7-014 Omitting Source Node Flag

When all of the following bundle processing control flag bits are set to FALSE, bundle source node ID of the bundle shall be omitted:

1. "Request reporting of bundle reception"
2. "Request reporting of bundle forwarding"
3. "Request reporting of bundle delivery"
4. "Request reporting of bundle deletion"

Status:	Draft
Source:	RFC 9171 Section 4.2.3
Rationale:	Control flag values are defined in RFC 9171 section 4.2.3
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from HDTN with all bundle processing control flag bits set to false and confirms the bundle source node ID has been omitted.

BPv7-015 Block Processing Control Flags

The block processing control flags shall be processed as a bit field.

Status:	Vetted
Source:	RFC 9171 Section 4.2.4
Rationale:	Bit fields are defined in the data dictionary. Bundle processing control flags assert properties of the bundle as a whole. They are conveyed in the primary block of the bundle. Control flag values are defined in RFC 9171 section 4.2.4 with details on page 15 and are required to be at least processed as bit field(s).
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when an inspection of the code shows the block processing control flag values as bit fields.

BPv7-016 The dtn URI Scheme-SSP

When the string value of a BP endpoint ID's Scheme-Specific Part (SSP) is equal to "none", the SSP shall be set to an unsigned integer value of zero.

Status:	Vetted
Source:	RFC 9171 Section 4.2.5.1.1
Rationale:	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. URI scheme code definition document requirements are defined in RFC 9171 section 4.2.5.1.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from HDTN with the endpoint ID of "dtn:none" and confirms the SSP has been set to zero.

BPv7-017 Node ID

The EID of a node's administrative endpoint shall uniquely identify that node; no two node endpoints are the same.

Status:	Vetted
Source:	RFC 9171 Section 4.2.5.2
Rationale:	Network management is necessary for determining the health and statistics of the network. The administrative endpoint of a node code definition document requirements are defined in RFC 9171 section 4.2.5.2.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when an inspection of the code shows that the EID of a node's administrative endpoint is generated to be unique to that node.

BPv7-018 Keep CBOR Encoding

The CBOR- encoded values of all fields in the primary block shall remain unchanged from point-to-point.

Status:	Vetted
Source:	RFC 9171 Section 4.3.1
Rationale:	Unchanging the primary block provides integrity and accountability of the network. Rule to keep the primary block unchanged is defined in RFC 9171 section 4.3.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that confirms HDTN does not modify the CBOR-encoded values of any field in the primary block of a bundle.

BPv7-019 Primary Header Field Order

The fields of the primary bundle block shall be in the following order when present: version, bundle processing control flags, CRC type, destination node EID, source node EID, report-to EID, creation timestamp, lifetime, fragment offset, total Application Data Unit Length, CRC.

Status:	Vetted
Source:	RFC 9171 Section 4.3.1
Rationale:	Primary bundle block fields explain what information is in what CBOR array. Field order of the primary bundle block is defined in RFC 9171 section 4.3.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from HDTN and confirms the primary block contents are in the specified order.

BPv7-020 Bundle Age

When Bundle Age Block extension block is present, Bundle Age shall be obtained from the Bundle Age Block extension block.

Status:	Draft
Source:	RFC 9171 Section 4.3.1
Rationale:	Bundle Age Block is specifically for the bundle's age and its computation to its expiration. Bundle Age Block details are defined in CCSDS 734.2-P-1.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from HDTN with a Bundle Age Block present and confirms the Bundle Age has been calculated from it.

BPv7-021 Fragment Offset

Fragment offset shall be present in the primary block when the bundle processing control flags of the primary block indicate that the bundle is a fragment.

Status:	Draft
Source:	RFC 9171 Section 4.3.1
Rationale:	Bundles can contain the full bundle or be a subset of the bundle, i.e., a fragment. Fragment Offset is defined in RFC 9171 section 4.3.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a fragmented bundle from HDTN and confirms the fragment offset is present.

BPv7-022 Canonical Block Field Order

The fields of every canonical block shall be listed in order and are: Block Type Code, Block Number, Block Processing Control Flags, CRC Type, Block Type Specific Data.

Status:	Vetted
Source:	RFC 9171 Section 4.3.2
Rationale:	Canonical bundle block(s) fields explain what information is in what CBOR array. Canonical Bundle Block is defined in RFC 9171 section 4.3.2. Note: When CRC is not present, they are omitted in the element count.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle with a canonical block from HDTN and confirms the canonical block fields are in the specified order.

BPv7-023 CRC Computation

CRC's computation shall be performed using a concatenation of all bytes of the block with the CRC's field temporarily set to zero as the binary dividend.

Status:	Vetted
Source:	RFC 9171 Section 4.3.1
Rationale:	CRC aids in integrity of the bundle since it is calculated by every piece in the header. The concatenation of all bytes of the block includes the CBOR "break" characters and the CRC field itself. CRC computation is defined in RFC 9171 section 4.3.1
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the computed CRC value is correct when the bytes of a CRC field is zero.

BPv7-024 Previous Node Block Format

Previous Node Block Node ID shall conform to Section 4.2.5.2 of RFC 9171.

Status:	Vetted
Source:	RFC 9171 Section 4.4.1
Rationale:	The Previous Node Block, block type 6, identifies the node that forwarded this bundle to the local node; its block-type-specific data is the node ID of the transmitter node. RFC 9171 defined Previous Node Block Node ID details in Section 4.4.1 and specifically calls out section 4.2.5.2.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that receives a bundle from HDTN and confirms the previous node block conforms to RFC 9171.

BPv7-025 Previous Node Block Limitation

The bundle shall contain a Previous Node Block after its first hop when Previous Node Block is enabled.

Status:	Vetted
Source:	RFC 9171 Section 4.4.1
Rationale:	There will not be a Previous Node Block when the bundle is first initialized. Previous Node Block details are defined in RFC 9171 section 4.4.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when enabling a Previous Node Block and verify the creation node bundle does not contain a Previous Node Block.

BPv7-026 Bundle Age Instances

When the bundle's creation time is zero, then the bundle shall contain an occurrence of a Bundle Age Block extension block.

Status:	Vetted
Source:	RFC 9171 Section 4.4.2
Rationale:	Bundle Age Block details are required in CCSDS 734.2-P-1.1; the bundle age units are defined in RFC 9171 section 4.4.2.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when the bundle's creation time is zero and contains a bundle age block.

BPv7-027 Forwarding Bundle Failure

The BPA shall declare failure in forwarding the bundle when any reason from the IANA "Bundle Status Report Reason Codes" registry is indicated.

Status:	Vetted
Source:	RFC 9171 Section 5.4.1
Rationale:	If the bundle did not send correctly, the system needs to know about it so it can correct it; a forwarding bundle failure is defined in RFC 9171 section 5.4.1.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when an inspection of the code shows a check to the IANA "Bundle Status Report Reason Codes" prior to forwarding the bundle.

BPv7-028 Forwarding Failed Procedure

Forwarding failed procedure shall be performed with Step 4 and Step 5 of Section 5.4 of RFC 9171 when Previous Node Block is enabled AND the bundle requires forwarding the bundle back to the previous node.

Status:	Vetted
Source:	RFC 9171 Section 5.4.2
Rationale:	When bundle tracking is on, then the bundle can use the tracking information to return back to the previous node. Details on this procedure are defined in RFC 9171 section 5.4.2.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends a bundle to HDTN that is unable to be forwarded and confirms the bundle is forwarded back to the sender.

BPv7-029 Bundle Expiration Calculation

The Bundle Age Extension block shall be used for the bundle expiration calculation when the bundle expiration is not using the timestamp.

Status:	Draft
Source:	RFC 9171 Section 5.5
Rationale:	If the bundle does not have the Bundle Age extension block, it may be calculated from the timestamp; not calculating the bundle expiration from the timestamp can be due to it being zero or the clock is untrustworthy; Bundle age is determined by subtracting the bundle's creation timestamp time from the current time; Bundle expiration is detailed in section 5.5 RFC 9171. DTN.6.10004
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when the Bundle Age Extension block calculates the bundle expiration calculation.

BPv7-030 Bundle Expiration Deletion

When a bundle is declared expired, the BPA shall set Status Report Reason Code to “Lifetime expired.”

Status:	Vetted
Source:	RFC 9171 Section 5.5
Rationale:	There is no need for the network to hold onto an expired bundle. Bundle expiration is detailed in section 5.5 RFC 9171.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test is performed that sends an expired bundle to HDTN and confirms the bundle has been deleted.

BPv7-031 Bundle Fragmentation Payload

When fragmenting a bundle, the payload shall be split between the fragments without overlapping.

Status:	Draft
Source:	RFC 9171 Section 5.8
Rationale:	Fragments of payloads cannot overlap to avoid data being duplicated, data corruption, and non-efficient space management for the network; Bundle fragmentation is defined in section 5.8 of RFC 9171.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows a fragmented bundle is split without overlapping.

BPv7-032 Bundle Fragmentation Concatenation

The concatenation of the payloads of all fragments produced by fragmentation shall be identical to the payload of the unfragmented bundle.

Status:	Draft
Source:	RFC 9171 Section 5.8
Rationale:	All payload fragments together must equal, in size and content, of the original payload; Bundle fragmentation is defined in section 5.8 of RFC 9171.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when reassembling a fragmented bundle's payload equals the original payload.

BPv7-033 Bundle Fragmentation Primary Block Flags

Bundle processing control flags within the primary block of a fragment shall indicate that the bundle is a fragment and provides the fragment offset and the total application data unit length.

Status:	Draft
Source:	RFC 9171 Section 5.8
Rationale:	Primary block stays the same for the fragments aside from the control flags pertaining to the specific fragment; Bundle fragmentation is defined in section 5.8 of RFC 9171.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a fragmented bundle's bundle processing control flags show the fragment offset and the total application data unit length.

BPv7-034 Bundle Fragmentation CRC

When fragmenting a bundle, the CRC of the primary block shall be recomputed for each fragment.

Status:	Draft
Source:	RFC 9171 Section 5.8
Rationale:	Bundles with CRCs will need to be recalculated if it is split into fragmented bundles. This is to ensure proper CRC values so the bundle fragment(s) aren't subject to being a "malformed bundle"; Bundle fragmentation is defined in section 5.8 of RFC 9171.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when an inspection of the code shows a recompilation of the CRC for each fragmented bundle.

BPv7-035 Bundle Fragmentation Zero Offset Fragment

When a fragmented bundle is the zero-offset fragment, all extension blocks of the fragmented bundle shall be replicated in the fragment.

Status:	Draft
Source:	RFC 9171 Section 5.8
Rationale:	Bundle fragmentation is defined in section 5.8 of RFC 9171.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a fragmented bundle is flagged to be a fragmented bundle and the offset is zero for the extension blocks of the fragmented bundle.

BPv7-036 Bundle Fragmentation Extension Blocks

Extension blocks of a fragmented bundle with a "Block must be replicated in every fragment" flag Bundle Processing Control Flag set to TRUE shall be replicated in every fragment.

Status:	Draft
Source:	RFC 9171 Section 5.8
Rationale:	Fragments just like bundles are subject to following the flags set; Bundle fragmentation is defined in section 5.8 of RFC 9171.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when the "Block shall be replicated in every fragment" flag is set to 1 in each fragment of a bundle.

BPv7-037 Reassembly for Payloads

The reassembled ADU shall replace the payload of the zero-offset fragment.

Status:	Draft
Source:	RFC 9171 Section 5.9
Rationale:	Reassembly is defined in section 5.9 of RFC 9171.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when reassembling a fragmented bundle's payload equals the original payload.

BPv7-038 Reassembly Label

The "Reassembly pending" retention constraint shall be removed from fragments with the same source node ID and creation timestamp as the original fragment.

Status:	Draft
Source:	RFC 9171 Section 5.9
Rationale:	Need to update the retention constraint to let the node know what is next to be done; Reassembly is defined in section 5.9 of RFC 9171.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when an inspection of the code shows the "Reassembly pending" retention constraint is removed from every fragment with the same source node ID and creation timestamp.

BPv7-039 Retention Constraints for Deletion

When a bundle is declared for deletion, there shall be no bundle retention constraints.

Status:	Vetted
Source:	RFC 9171 Section 5.1
Rationale:	When discarding a bundle, there is no need for retention constraints since there will be no retention; Retention constraints for bundle deletions are defined section 5.10 in RFC 9171.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when an inspection shows that when a bundle is declared for deletion there are not bundle retention constraints.

BPv7-040 Convergence-Layer Services

Convergence-layer protocols shall provide rate limiting, congestion control or both.

Status:	Vetted
Source:	RFC 9171 Section 7.2
Rationale:	Some convergence-layer protocols do not have the capabilities to handle large throughputs; convergence-layer protocols are defined in section 7.2 of RFC 9171. HDTN only provides rate limiting convergence-layer protocols.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when an inspection shows that the code is performing rate limiting.

BPv7-041 Security

When requested, security shall use BPSec.

Status:	Vetted
Source:	RFC 9171 Section 8
Rationale:	The data in the network must keep its integrity, encryption, and authentication when required; security considerations are defined in section 8 of RFC 9171.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that BPSec is used when requested.

3 BPSEC REQUIREMENTS

BPSec-001 Security Operation Uniqueness

BPSec shall apply one security service to a security target per bundle.

Status:	Vetted
Source:	RFC 9172 Section 3.2
Rationale:	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.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when the user configuration of two same security operations for the same target block is not allowed and generates an error.

BPSec-002 Security Contexts

BPSec shall support the default security contexts in the approved security contexts registry: BCB-AES-GCM and BIB-HMAC-SHA2.

Status:	Vetted
Source:	RFC 9172 Section 2.4
Rationale:	HDTN currently implements all default security contexts as per RFC 9173. These are the minimum security contexts that need to be supported for interoperability.
Verification Method:	Test
Verification Approach:	This requirement is considered verified when the default integrity and confidentiality security contexts BCB-AES-GC and BIB-HMAC-SHA2 are both part of the options security context parameters in the BPSec config file and are working as expected.

BPSec-003 Target Multiplicity

BPSec shall utilize a single security block to represent multiple security operations.

Status:	Vetted
Source:	RFC 9172 Section 3.3
Rationale:	Reducing the number of security blocks in the bundle reduces the amount of redundant information in the bundle.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when code inspection shows that BPSec security blocks (Bpv7AbstractSecurityBlock) support multiple results per target.

BPSec-004 Target Identification

BPSec shall set a security target value in a security block to the block number of the target block.

Status:	Vetted
Source:	RFC 9172 Section 3.4
Rationale:	A security target must be uniquely and unambiguously identifiable when processing a security block. Block number field in the extension block should be used for this purpose. Placing the set of target blocks covered by operations in an easily accessible field allows Bundle Protocol Agents (BPAs) to quickly scan this field to assess whether new Security Operations (SOps) can be added to a bundle and whether the block has operations that must be processed.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when displaying the value of the block to which the security operation is applied matches the security target field in the corresponding security extension block.

BPSec-005 BIB Block Type

The block-type-specific data field of a BIB shall follow the structure of the ASB.

Status:	Vetted
Source:	RFC 9172 Section 3.7
Rationale:	All security blocks share the same block-type-specific data structure as these blocks have common aspects.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when the inspection of a bundle with BIB block on Wireshark shows that its block-type-specific data structure follows the structure of the ASB.

BPSec-006 BIB Security Targets

A security target of a BIB shall reference blocks excluding BIB or BCB.

Status:	Vetted
Source:	RFC 9172 Section 3.7
Rationale:	An appropriate target block that a BIB should be able to reference is any block that may have its block-type-specific-data signed.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when the user is not allowed to add a BIB with the security target as BIB or BCB.

BPSec-007 BIB Integrity Mechanism

The security context shall utilize either:

1. Authentication mechanism.

2. An error detection mechanism.

Status:	Vetted
Source:	RFC 9172 Section 3.7
Rationale:	The integrity mechanism used by the BIB is given by the security context associated with the BIB and may represent either signed or unsigned integrity. In this way, BIB can be used to represent authentication (with a signed integrity mechanism) or simply error detection (with an unsigned integrity mechanism).
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test successfully authenticates a security source using unsigned BIB or signed BIB.

BPSec-008 BCB Replication in Bundle Fragments

BPSec shall set the flag ‘Block must be replicated in every fragment’ to true for a BCB which has payload as a target.

Status:	Draft
Source:	RFC 9172 Section 3.8
Rationale:	BCB in each fragment indicates to a receiving node that the payload portion of each fragment represents ciphertext.
Verification Method:	Test
Verification Approach:	NA

BPSec-009 BCB Processing

BPSec shall set the flag ‘Block must be removed from bundle if it cannot be processed’ to false for a BCB.

Status:	Vetted
Source:	RFC 9172 Section 3.8
Rationale:	The removal of a BCB from a bundle has significant consequences since the BCB is the sole indication in the bundle that the BCB target block(s) have had their block-type-specific-data field encrypted. Removing a BCB would make it impossible for future BPAs to decrypt the block.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when code inspection shows that BCB has " Block must be removed from bundle if it cannot be processed’ flag set to false.

BPSec-010 BCB Block Type

The block-type-specific data fields of a BCB shall follow the structure of the ASB.

Status:	Vetted
Source:	RFC 9172 Section 3.8
Rationale:	All security blocks share the same block-type-specific data structure as these blocks have common aspects.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when the inspection of a bundle with BCB block on Wireshark shows that its block-type-specific data structure follows the structure of the ASB.

BPSec-011 BCB Security Targets

A security target of a BCB shall reference either:

1. The payload block.

2. A non-security extension block.

3. A BIB.

Status:	Vetted
Source:	RFC 9172 Section 3.8
Rationale:	An appropriate target block that a BCB should be able to reference is any block that may have its block-type-specific-data encrypted. Only payload, non-security extension blocks and BIB blocks are allowed to be encrypted for proper processing of bundles. A BCB MUST NOT include another BCB as a security target as other BCBs in a bundle cannot be encrypted. Doing so would hide the fact that other blocks in a bundle have been encrypted and removes the ability to decrypt them.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when the user is allowed to add any of the following blocks as a security target of BCB the payload block, a non-security extension block or a BIB and is not allowed to include another BCB as a security target.

BPSec-012 BCB Disallowed Security Targets 1

A BCB shall target blocks excluding the primary block.

Status:	Vetted
Source:	RFC 9172 Section 3.8
Rationale:	Encrypting the primary block hides bundle identity.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when the user is not allowed to add the payload block as a target for a BCB.

BPSec-013 BCB Disallowed Security Targets 2

A BCB shall target a BIB only when sharing a security target with that BIB.

Status:	Vetted
Source:	RFC 9172 Section 3.8
Rationale:	Encrypting a BIB and not the target block of the BIB removes the ability to check the integrity of that target block.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when the user is allowed to add BIB as a security target for a BCB only if the BCB shares a security target with that BIB.

BPSec-014 Authenticated Encryption with Associated Data

A BCB shall utilize a confidentiality cipher that provides Authenticated Encryption with Associated Data (AEAD).

Status:	Vetted
Source:	RFC 9172 Section 3.8
Rationale:	This allows confirming that ciphertext, block processing flags and other blocks in the bundle have not been modified.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test of confidentiality using an authentication tag successfully verifies that ciphertext and additional specified blocks in the bundle have not been modified.

BPSec-015 Authentication Tag Placement

Additional information created by the cipher suite shall be placed either in:

1. The security result field.

2. The generated ciphertext.

Status:	Vetted
Source:	RFC 9172 Section 3.8
Rationale:	Security contexts used by the BCB must specify whether the authentication tag is included in the BCB as a security result or whether it is represented with the ciphertext that replaces the security target block's plaintext.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the user is able to configure an authentication tag to be included either in the BCB as a security result or in the ciphertext that replaces the security target block's plaintext.

BPSec-016 Encryption in Place

When applying a BCB, BPSec shall encrypt the security target body data ‘in-place’.

Status:	Vetted
Source:	RFC 9172 Section 3.8
Rationale:	This eliminates the need to move the data making the encryption faster and more reliable and provides better performance.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when an inspection of the code of encryption function shows that encryption was performed in place.

BPSec-017 BCB and BIB Blocks Interactions 1

When adding a BCB to a bundle with matching security targets of an existing BIB targets, then BPSec shall encrypt the existing BIB.

Status:	Vetted
Source:	RFC 9172 Section 3.9
Rationale:	This is needed to handle the special case of protecting the plaintext integrity of the target block when that plaintext had been replaced by ciphertext.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that when adding a BCB to a bundle, if some (or all) of the security targets of the BCB match all of the security targets of an existing BIB, then the existing BIB is also be encrypted.

BPSec-018 BCB and BIB Blocks Interactions 2

When adding a BCB to a bundle with matching security targets of an existing BIB security targets, then BPSec shall remove the security results associated with the BCB security targets from that BIB and place them in a new encrypted BIB.

Status:	Vetted
Source:	RFC 9172 Section 3.9
Rationale:	This processing rule prevents security information about the unencrypted target block from persisting after the target block has been encrypted.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified 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 any security results in the BIB associated with the BCB security targets is removed from the BIB and placed in a new encrypted BIB.

BPSec-019 BCB and BIB Blocks Interactions 3

BPSec shall prevent a BIB addition for a security target that is already the security target of a BCB.

Status:	Vetted
Source:	RFC 9172 Section 3.9
Rationale:	This is needed to not cause ambiguity in block processing order.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when the user is prevented from adding a BIB for a security target that is already the security target of a BCB.

BPSec-020 BCB and BIB Blocks Interactions 4

BPSec shall check a BIB integrity value when the BIB data is unencrypted.

Status:	Vetted
Source:	RFC 9172 Section 3.9
Rationale:	A BIB integrity value MUST NOT be checked if the BIB is the security target of an existing BCB as in this case, the BIB data is encrypted.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that when BIB is the security target of an existing BCB, that BIB is encrypted and not checked when processing the security result.

BPSec-021 BCB and BIB Blocks Interactions 5

BPSec shall check a BIB integrity value when the corresponding security target data is unencrypted.

Status:	Vetted
Source:	RFC 9172 Section 3.9
Rationale:	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 ciphertext as it has been encrypted.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that when BIB has the same security target as a BCB, that BIB is not checked.

BPSec-022 BCB and BIB Blocks Interactions 6

BPSec shall prevent a BIB from having a BCB for its security target.

Status:	Vetted
Source:	RFC 9172 Section 3.9
Rationale:	An appropriate target block that a BIB should be able to reference is any block that may have its block-type-specific-data signed.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when the user is prevented from adding a BCB as a security target for a BIB.

BPSec-023 Canonical Form of the Primary Block

Concise Binary Object Representation (CBOR) values from the primary block shall be canonicalized using the rules for Deterministically Encoded CBOR.

Status:	Vetted
Source:	RFC 9172 Section 4
Rationale:	Canonicalization algorithms transcode the contents of a security target into a canonical form. Security services require consistency and determinism in how information is presented to cipher suites at security sources, verifiers, and acceptors.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when an inspection of the code and bundle with BPSec enabled shows that the primary block is canonicalized using the rules for Deterministically Encoded CBOR.

BPSec-024 Non-Primary Block Structure

Non-primary blocks shall share the same block structure.

Status:	Vetted
Source:	RFC 9172 Section 4
Rationale:	Security services require consistency and determinism in how information is presented to cipher suites at security sources, verifiers, and acceptors.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when code inspection shows that all non-primary blocks share the same block structure.

BPSec-025 Canonical Form of the Non-Primary Block

Non-primary blocks shall be canonicalized.

Status:	Vetted
Source:	RFC 9172 Section 4
Rationale:	To ensure that 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.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when an inspection of the bundle with BPSec enabled on Wireshark shows it has all its non-primary blocks canonicalized.

BPSec-026 Canonical Form Non-Primary Block CBOR Values

CBOR values from the non-primary block shall be canonicalized using the rules for Deterministically Encoded CBOR.

Status:	Vetted
Source:	RFC 9172 Section 4
Rationale:	Security services require consistency and determinism in how information is presented to cipher suites.
Verification Method:	Test and Inspection
Verification Approach:	This requirement will be considered verified when an inspection of the code and bundle shows that CBOR values from the non-primary block shall be canonicalized using the rules for Deterministically Encoded CBOR.

BPSec-027 Canonical Form

Only the block-type-specific data field shall be provided to a cipher suite for encryption.

Status:	Vetted
Source:	RFC 9172 Section 4
Rationale:	BPSec operates on data fields within bundle blocks (e.g., the block- type-specific data field). In their canonical form, these fields shall include their own CBOR encoding and not any other encapsulating CBOR encoding.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when code inspection of the encryption crypto function shows that only block-type-specific data field is provided as input for encryption.

BPSec-028 Canonical Form of the Other Blocks Encryption

Only the block-type-specific data within a non-primary block shall be encrypted and included in the canonical form used by the cipher suite for encryption.

Status:	Vetted
Source:	RFC 9172 Section 4
Rationale:	BPSec operates on data fields within bundle blocks (e.g., the block- type-specific data field). In their canonical form, these fields shall include their own CBOR encoding and not any other encapsulating CBOR encoding.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a Wireshark inspection of bundle with BCB only has the block-type-specific data within a non-primary block encrypted and included in the canonical form used by the cipher suite for encryption.

BPSec-029 Canonical Form of the Other Blocks Decryption

Only the block-type-specific data within a non-primary block shall be decrypted and included in the canonical form used by the cipher suite for decryption.

Status:	Vetted
Source:	RFC 9172 Section 4
Rationale:	BPSec operates on data fields within bundle blocks (e.g., the block- type-specific data field). In their canonical form, these fields shall include their own CBOR encoding and not any other encapsulating CBOR encoding.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a Wireshark inspection of bundle with BCB only has the block-type-specific data within a non-primary block decrypted and included in the canonical form used by the cipher suite for decryption.

BPSec-030 Associated Authenticated Data

BPSec shall apply an integrity-protection mechanism to non block-type-specific data fields within a nonprimary block.

Status:	Vetted
Source:	RFC 9172 Section 4
Rationale:	This allows confirming that other blocks in the bundle have not been modified.
Verification Method:	Test and Inspection
Verification Approach:	This requirement will be considered verified when a test shows that when authentication tag is configured, an integrity-protection mechanism is applied to those additional fields.

BPSec-031 Receiving BCBs 1

When a received bundle contains a BCB, the receiving node shall determine whether it is the security acceptor for any of the security operations in the BCB.

Status:	Vetted
Source:	RFC 9172 Section 5.1.1
Rationale:	This is because the acceptor does the final processing of the security operation and is the only node which decrypt and process the security operations in the BCB.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that there's a check for whether the node is a security acceptor before processing any security operations in the BCB.

BPSec-032 Receiving BCBs 2

BPSec shall process the targets according to the security policy after a failed confidentiality security operation.

Status:	Vetted
Source:	RFC 9172 Section 5.1.1
Rationale:	In case of security operation failure, the action for target processing is based on the security policy configured by the user.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that if a node that is the security acceptor or verifier for any of the security operations in the BCBs, and the security verification fails, the target shall be processed as specified in the BPSec config file.

BPSec-033 Receiving BCBs 3

BPSec shall generate a bundle status report indicating the failure of a confidentiality security operation.

Status:	Vetted
Source:	RFC 9172 Section 5.1.1
Rationale:	In case of security operation failure, a bundle status report generation will help for debugging and tracking these failures.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that if a node that is the security acceptor or verifier for any of the security operations in the BCBs, and the security verification fails, a bundle status report with a failure reason code is generated.

BPSec-034 Receiving BCBs 4

When the receiving node is the destination of the bundle, the node shall decrypt BCBs remaining in the bundle.

Status:	Vetted
Source:	RFC 9172 Section 5.1.1
Rationale:	The bundle destination is, by necessity, the acceptor of any block remaining in the bundle. Since a bundle will no longer exist after processing at this BPA, all blocks need to be accepted prior to passing the bundle payload to applications resident on the destination BPA.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that if a node that is the bundle final destination receives a bundle which has multiple BCBs, it successfully verifies, decrypts all ciphertext and removes all BCBs from the bundle.

BPSec-035 Receiving BCBs 5

When the receiving node is not the destination of the bundle, the node shall process the BCB in accordance with the security policy.

Status:	Vetted
Source:	RFC 9172 Section 5.1.1
Rationale:	If the receiving node is a verifier, it shall process the BCB based on the configured security policy.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test has a verifier node that is able to successfully process the BCBs according to the security policy in the BPSec config file.

BPSec-036 Receiving BCBs 6

When the security policy of a node requires a node to apply confidentiality to a specific security target and no such BCB is present in the bundle, the node shall process this security target in accordance with the security policy.

Status:	Vetted
Source:	RFC 9172 Section 5.1.1
Rationale:	HDTN removes the security target from the bundle because the confidentiality (and possibly the integrity) of the security target cannot be guaranteed.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows 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 security target is removed from the bundle.

BPSec-037 Receiving BCBs 7

When the security processing results in the removal of the payload block BPSec shall discard the bundle.

Status:	Vetted
Source:	RFC 9172 Section 5.1.1
Rationale:	The payload is required in a bundle.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test where BCB processing results in payload block removal results in discarding the bundle.

BPSec-038 Receiving BCBs 8

The bundle shall be discarded when BPSec fails to decrypt the encrypted payload block.

Status:	Vetted
Source:	RFC 9172 Section 5.1.1
Rationale:	This is a security failure meaning the bundle could be compromised. However, the payload block is required in a bundle so the whole bundle is discarded in this case.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that when at the receiving security verifier or acceptor node an encrypted payload block cannot be decrypted, the bundle is discarded and processed no further (we use incorrect key to test this scenario).

BPSec-039 Receiving BCBs 9

When an encrypted security target other than a payload block cannot be decrypted, then BPSec shall discard the target and the corresponding security blocks.

Status:	Vetted
Source:	RFC 9172 Section 5.1.1
Rationale:	This is a security failure meaning the bundle could be compromised, so the corresponding target and security blocks should be discarded.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that when at the receiving security verifier or acceptor node when 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 shall be discarded and processed no further.

BPSec-040 Receiving BCBs 10

When the security block is deleted from a bundle, BPSec shall generate a status report showing block deletion.

Status:	Vetted
Source:	RFC 9172 Section 5.1.1
Rationale:	Status report helps for debugging and reflecting the exact root cause of the failure.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that If a block is deleted from a bundle as a result of security operation, a status reports is generated to reflect block deletion.

BPSec-041 Receiving BCBs 11

When a bundle is dropped following a BCB processing failure, BPSec shall generate a status report showing bundle deletion.

Status:	Vetted
Source:	RFC 9172 Section 5.1.1
Rationale:	Status report helps for debugging and reflecting the exact root cause of the failure.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that If a bundle is deleted as a result of security operation, a status reports is generated to reflect block deletion.

BPSec-042 Receiving BCBs 12

When BPSec decrypts a BCB, the recovered plaintext for the security targets shall replace the ciphertext in the corresponding security targets' block-type specific data fields.

Status:	Vetted
Source:	RFC 9172 Section 5.1.1
Rationale:	The final bundle received by the application should only have decrypted data.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a bundle inspection in Wireshark at the acceptor node for a confidentiality test shows that the ciphertext in the bundle was replaced with the correct plain text for the security target block-type specific data field.

BPSec-043 Receiving BCBs 13

When the plaintext is of a different size than the ciphertext, BPSec shall update the framing of the CBOR byte string of this field.

Status:	Vetted
Source:	RFC 9172 Section 5.1.1
Rationale:	This will ensure this field remains a valid CBOR byte string.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a bundle inspection in Wireshark at the acceptor node for a confidentiality test shows that if the plain text is of a different size than the ciphertext, the framing of the CBOR byte string is updated.

BPSec-044 Receiving BCBs 14

When a BCB contains multiple security operations, each operation processed by the node shall be treated as if the security operation has been represented by a single BCB with a single security operation.

Status:	Vetted
Source:	RFC 9172 Section 5.1.1
Rationale:	This helps for the purposes of report generation and policy processing.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when the inspection of the code shows that if the BCB data structure contains multiple security operations, each operation processed by the node shall be treated as if the security operation has been represented by a single BCB with a single security operation.

BPSec-045 Receiving BIBs 1

When a received bundle contains a BIB, the receiving node shall determine whether it is the security acceptor for any of the security operations in the BIB.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	This check is needed to determine the steps to process the bundle.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that for a received bundle by HDTN ingress module with a BIB, there's a check in the code to determine whether the node is the security acceptor for any of the security operations in the BIB.

BPSec-046 Receiving BIBs 2

When the receiving node is the security acceptor for any of security operations in a BIB:

1. The node shall process those operations.

2. Remove any operation specific information from the BIB prior to forwarding the bundle.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	This is because the acceptor does the final processing of the security operation.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that If the receiving node is configured as the security acceptor for any security operations in a BIB, the node successfully processes those operations and removes any operation specific information from the BIB prior to delivering data to an application at the node or forwarding the bundle.

BPSec-047 Receiving BIBs 3

BPSec shall process the targets according to the security policy following an integrity security operation failure.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	In case of security operation failure, the action for target processing is based on the security policy configured by the user.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that when receiving a bundle with BIB and if the processing a security operation fails, the target is processed according to the security policy in the BPSec config file.

BPSec-048 Receiving BIBs 4

BPSec shall generate a bundle status report to indicate an integrity security operation failure.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	In case of security operation failure, a bundle status report generation will help for debugging and tracking these failures.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that when receiving a bundle with BIB and if processing of a security operation fails, a bundle status report indicating the failure is generated.

BPSec-049 Receiving BIBs 5

When BPSec removes all the security operations for a given BIB, that BIB shall be removed from the bundle.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	This is the final stage in the security operation lifecycle and that BIB is not needed anymore.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that when all security operations for a BIB have been removed from the BIB, the BIB is removed from the bundle.

BPSec-050 Receiving BIBs 6

BPSec shall process a BIB only if the security target of the BIB is not the security target of a BCB in the bundle.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	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; therefore, the BIB cannot be verified until the security target has been decrypted by processing the BCB.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows with a bundle where both a BIB and a BCB share a security target, that BIB is not processed until the security target has been decrypted by processing the BCB.

BPSec-051 Receiving BIBs 7

When the security policy of a node requires a node to apply integrity to a specific security target and no such BIB is present in the bundle, the node shall process this security target in accordance with the security policy.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	We shall process the BIB based on the configured security policy.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when we emulate a test where the security policy of a node specifies a node should have applied integrity to a specific security target but no such BIB is present in the bundle, it results in the node processing this security target as configured in the BPSec config file.

BPSec-052 Receiving BIBs 8

When the security policy of a node specifies a node should have applied integrity to a specific security target and no such BIB is present in the bundle, the node shall remove the security target from the bundle when the security target is not the payload or primary block.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	This is considered an integrity security failure and security target could be compromised and should be removed. The payload and primary blocks are required in a bundle and cannot be removed.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that if the security policy of a node specifies a node should have applied integrity to a specific security target and no such BIB is present in the bundle, the node removes the security target from the bundle as long as the security target is not the payload or primary block.

BPSec-053 Receiving BIBs 9

When the target block of the failed integrity security operation is the primary block, BPSec shall discard the bundle.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	The primary block is required in a bundle and cannot be discarded.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test with an integrity security operation targeting the primary block fails, the bundle is discarded.

BPSec-054 Receiving BIBs 10

When the target block of the failed integrity security operation is the payload block, BPSec shall discard the bundle.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	The payload block is required in a bundle and cannot be discarded.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test with an integrity security operation targeting the payload block fails, the bundle is discarded.

BPSec-055 Receiving BIBs 11

When a receiving node is not the security acceptor of a security operation in a BIB, the node shall attempt to verify the security operation.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	This is to prevent forwarding corrupt data.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that when a node is the security verifier of a security operation in a BIB, the node attempts to verify the security operation.

BPSec-056 Receiving BIBs 12

When a verification security operation fails, the node shall process the security target in accordance with local security policy.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	In case of security operation failure the action for target processing is based on the security policy configured by the user.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that if there's a BIB security operation failure, the node processes the security target in accordance with security policy configured in the BPSec config file.

BPSec-057 Receiving BIBs 13

When a payload integrity check fails at a waypoint, BPSec shall trigger a failure of a payload integrity check at the bundle destination.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	The waypoint shall verify the security operation and prevent forwarding corrupt data.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that if a payload integrity check fails at a waypoint node, the payload is processed as a failure of a payload integrity check at the bundle destination.

BPSec-058 Receiving BIBs 14

When a BIB integrity check passes at a waypoint, the node shall retain the security operation in the BIB prior to forwarding.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	If the integrity check passes at the waypoint, the bundle is verified and should be forwarded to the next hop. Only the acceptor removes the BIB extension block.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that if a BIB integrity check passes at a waypoint, the node will retain the security operation in the BIB prior to forwarding.

BPSec-059 Receiving BIBs 15

When a BIB contains multiple security operations, each operation processed by the node shall be treated as if the security operation has been represented by a single BIB with a single security operation.

Status:	Vetted
Source:	RFC 9172 Section 5.1.2
Rationale:	This is for the purposes of report generation and policy processing.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when the inspection of the code shows that if the BIB data structure contains multiple security operations, each operation processed by the node shall be treated as if the security operation has been represented by a single BIB with a single security operation.

BPSec-060 Bundle Fragmentation and Reassembly

When fragmentation is required for a bundle payload, and BPSec has applied security services to that bundle, BPSec shall only fragment the payload block.

Status:	Draft
Source:	RFC 9172 Section 5.2
Rationale:	This is because security blocks, like all extension blocks, can never be fragmented.
Verification Method:	Test
Verification Approach:	NA - HDTN Bundle fragmentation for BPv7 is not implemented at this time. This will be added soon.

BPSec-061 BIB and BCB Blocks Addition to a Bundle

BPSec shall prevent the addition of BCB or BIB to a bundle if the ‘Bundle is a fragment’ flag is set to true in the bundle processing control flags field.

Status:	Vetted
Source:	RFC 9172 Section 5.2
Rationale:	This is because fragmentation delays certain security processing until bundle reassembly, which results in some extension blocks being duplicated, and the result of security failures being ambiguous in the presence of fragmentation.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when BIB or BCB are not allowed to be added to a bundle if the "Bundle is a fragment" flag set - HDTN Bundle fragmentation for BPv7 is not implemented at this time. This will be added soon.

4 ROUTING REQUIREMENTS

Routing-001 Contact Plan Modification

All route lists shall be recomputed when the contact plan has been modified.

Status:	Vetted
Source:	SABR 3.2.3.1
Rationale:	Contact plan changes may invalidate any or all earlier route computations. When the contact plan is updated, new routes are calculated.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that updates the HDTN contact plan and confirms the route list has been recomputed.

Routing-002 Expired Contacts

Expired contacts shall be deleted from the contact graphs.

Status:	Vetted
Source:	SABR 3.2.3.2
Rationale:	These contacts have exceeded their defined period of validity and are no longer used.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that loads a contact plan with expired contacts into HDTN and confirms the expired contacts have been removed from the contact plan.

Routing-003 Route Computation

The route with the earliest arrival time shall be selected from the list of candidate routes.

Status:	Vetted
Source:	SABR 3.2.4
Rationale:	Both CGR and CMR use the earliest time of arrival to select the best route.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that loads a contact plan into HDTN and verifies the route with the earliest arrival time has been selected from the list of candidate routes.

Routing-004 CGR Preparation

The list of candidate routes shall be computed from the contact plan.

Status:	Vetted
Source:	SABR 3.2.5
Rationale:	The contact plan is the schedule of contacts for each node. It is the input to the router. The list of candidate routes is computed for each final destination based on the root contact, final destination, and contact start and stop times. All routes are computed in order to select the best.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that loads a contact plan into HDTN and verifies the computed candidate routes match the expected values.

Routing-005 Routing Algorithm Selection

The routing algorithm shall be selected from one of the following: contact graph routing or contact multigraph routing.

Status:	Vetted
Source:	NA
Rationale:	HDTN is intended to support multiple routing algorithms including CGR, CMR, and others. The algorithm is selected by the user.
Verification Method:	Inspection
Verification Approach:	This requirement shall be considered verified when an inspection of the code shows that CGR and CMR are both supported as routing algorithms.

Routing-006 Rerouting Around Failed Node

The router shall select the route with the next earliest arrival time if it detects that the current route has failed.

Status:	Vetted
Source:	NA
Rationale:	All routes have been precomputed so that if the current route fails, the next best route can be used.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that simulates a route failure on the selected route and confirms that HDTN selects a new route.

5 TCPCL REQUIREMENTS

TCPCL-001 Network Byte Order

The integer encodings shall transmit in network byte order.

Status:	Vetted
Source:	RFC 9174 Section 2.1
Rationale:	This is the byte order for integers per RFC 9174. Big endian byte ordering is the same as network byte order. This means the bits are transmitted in this order: bits 0-7 first, then bits 8-15, then 16-23 and bits 24-31 last.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test shows that integer encodings are received in network byte order.

TCPCL-002 Session Establishment

When using TCPCL for bundle transmissions, communicating entities shall establish a TCPCL session.

Status:	Vetted
Source:	RFC 9174 Section 4
Rationale:	TCPCL requires a handshake with the other node to make certain both sides are compatible for transmission and reception.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test and inspection on Wireshark show two TCPCL nodes establishing a session and exchanging data successfully.

TCPCL-003 Contact Header

To establish a TCP connection, the active entity shall transmit its Contact Header.

Status:	Vetted
Source:	RFC 9174 Section 4.1
Rationale:	The contact header of TCPCL contains the required pieces for the two nodes to attach. Without sending the contact header, then no communication/handshake can partake.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified after a test is performed showing the active entity transmitting its Contact Header.

TCPCL-004 Contact Header Reception

When the passive entity receives the Contact Header of an active entity, then the passive entity shall transmit its Contact Header.

Status:	Vetted
Source:	RFC 9174 Section 4.1
Rationale:	To complete a handshake, the passive/receiver node must acknowledge reception of the active/transmitter node's contact header by sending out the passive node's contact header. By checking each node's contact header, each node is able to make certain a communication can be made.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test and inspection on Wireshark show that the passive entity transmitting its Contact Header after receiving the active entity's Contact Header.

TCPCL-005 TCP Idle Timeout

The TCP connection shall close when the entity timer matches the specified timeout waiting time.

Status:	Vetted
Source:	RFC 9174 Section 4.1
Rationale:	Timeouts avoid an idle network. Timeouts ensure the network will not be 'hung up' in any particular node.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that as soon as the timeout timer is reached the TCP connection closes.

TCPCL-006 TCP FIN

The entity shall use the TCP FIN mechanism when closing a TCP connection.

Status:	Vetted
Source:	RFC 9174 Section 4.1
Rationale:	There are two mechanisms within TCP for opening and closing connections: RST and FIN. FIN is the standard clean close for a TCP connection.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when an entity is closing a TCP connection uses the TCP FIN mechanism.

TCPCL-007 Enable TLS by CAN_TLS Flag

TLS shall change the CAN_TLS flag within its Contact Header to 1 when enabled.

Status:	Vetted
Source:	RFC 9174 Section 4.2
Rationale:	TLS is designed to facilitate privacy and data security for communications over the network. CAN_TLS indicates that the sending peer has enabled TLS security. It is recommended to enable TLS for all sessions.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that TLS is enabled by setting the following fields in the HDTN outduct config: tryUseTls, tlsIsRequired, useTlsVersion1_3 or useTlsVersion1_4, doX509CertificateVerification, verifySubjectAltNameInX509Certificate, certificationAuthorityPemFileForVerification.

TCPCL-008 Magic String Validation

The connection shall terminate when the "dtn!" string is not present in the contact header.

Status:	Vetted
Source:	RFC 9174 Section 4.3
Rationale:	Magic is 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). Magic is one of the 3 fields within a TCP contact header.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows, that upon receiving a message without the "dtn!" string in its contact header, the receiving entity terminates the connection.

TCPCL-009 Version Number Negotiation

The active entity shall terminate the TCP connection when the passive entity's TCPCL protocol version is lower than the active entity's version provided in the Contact Header.

Status:	Vetted
Source:	RFC 9174 Section 4.3
Rationale:	Both Contact Headers of a successful contact negotiation must have identical TCPCL version numbers. TCPCL versions are not backwards and forwards compatible.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the active entity terminates the TCP connection when the active entity receives a TCPCL protocol version lower than its version provided in the Contact Header.

TCPCL-010 Reattempt Failed Connections

Failed connection attempts shall use a (binary) exponential backoff mechanism to increase the delay between retries in the case of repeated failures.

Status:	Draft
Source:	RFC 9174 Section 4.1
Rationale:	The TCPCL entity should not overwhelm its target with repeated connection attempts.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that the backoff mechanism increases the delay between retries via a binary exponential when there are failed connection attempts.

TCPCL-011 Version Mismatch Reason Code

When the active entity terminates the TCP connection from TCPCL protocol version mismatch, then the passive entity shall terminate the TCP connection with a reason code of "Version Mismatch".

Status:	Vetted
Source:	RFC 9174 Section 4.3
Rationale:	Both Contact Headers of a successful contact negotiation must have identical TCPCL version numbers. TCPCL versions are not backwards and forwards compatible.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that the active entity terminated the TCP connection from version mismatch with a reason code of "Version Mismatch" when the TCPCL protocol version in the contact header is not supported.

TCPCL-012 TLS Lifetime

The lifetime of the underlying TCP connection shall match the lifetime of a TLS connection.

Status:	Vetted
Source:	RFC 9174 Section 4.4
Rationale:	When the TCP connection is terminated, then the TLS connection will also be terminated. This is because TLS resides on the TCP connection when it is enabled.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test with TLS enabled shows that the lifetime of the underlying TCP connection matches the lifetime of the TLS connection during the handshake.

TCPCL-013 Node ID Matching

The implementation shall terminate the session when a contact header contains a NODE-ID that does not match the URI "ipn:nextHopNodeId.0".

Status:	Vetted
Source:	RFC 9174 Section 4.4.1
Rationale:	ION 3.7.2 source code tcpcli.c line 1199 uses service number 0 for contact header. Therefore for SSL verification, implementations need to do a string compare of "ipn:nextHopNodeId.0". This is for compatibility between network implementations. If the validation result fails or if the result is absent and the security policy requires an authenticated node ID then the session will terminate (reason code: "Contact Failure").
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test proving that the entity will terminate the session when a contact header is received without the URI "ipn:nextHopNosdeId.0".

TCPCL-014 Certificate Profile

End-entity certificates used by a TCPCL entity shall conform to RFC 5280, or any updates or successors to that profile.

Status:	Draft
Source:	RFC 9174 Section 4.4.2
Rationale:	Certificates need to interoperable with other entities and therefore need to conform to those certificate profiles.
Verification Method:	Inspection
Verification Approach:	This requirement shall be considered verified when a code inspection shows that the TCPCL end-entity certificates conform to RFC 5280.

TCPCL-015 Version 3 Certificates

TCPCL shall require TCPCL version 3 certificates.

Status:	Vetted
Source:	RFC 9174 Section 4.4.2
Rationale:	The TCPCL requires version 3 certificates due to the extensions used by the TCPCL certificate profile. It will reject version 1 and version 2 end-entity certificates.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is provided showing TCPCL version 3 certificates are used.

TCPCL-016 Node ID Certificate

The TCPCL end-entity certificate shall contain a NODE-ID when applicable by CA policy.

Status:	Vetted
Source:	RFC 9174 Section 4.4.2
Rationale:	When assigned one or more stable DNS names, a TCPCL end-entity certificate contain a DNS-ID that authenticates those (fully qualified) names. When assigned one or more stable network addresses, a TCPCL end-entity certificate contains an IPADDR-ID that authenticates those addresses.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test shows a TCPCL end-entity certificate containing a NODE-ID when CA policy is enabled.

TCPCL-017 Type-ID

The active entity shall terminate the session if the type-id id-on-bundleEID does not match the passive entity's next hop endpoint ID.

Status:	Vetted
Source:	RFC 9174 Section 4.4.2.1.
Rationale:	Network integrity is lost when messages are delivered to the wrong node.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a bundle with a bad certificate is sent and is rejected at the receiving node.

TCPCL-018 TLS Authentication

The requested TLS handshake shall perform the TLS authentication.

Status:	Vetted
Source:	RFC 9174 Section 4.4.4
Rationale:	It is recommended to enable TLS for all sessions. TLS designed to facilitate privacy and data security for communications over the network.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that the TLS authentication happens during the TLS handshake.

TCPCL-019 Node ID Mismatch

When the active entity's SESS_INIT received differs from the intended node ID, the TCPCL session shall reject the SESS_INIT.

Status:	Vetted
Source:	RFC 9174 Section 4.6
Rationale:	Network integrity is lost when messages are delivered to the wrong node.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a test and bundle inspection on Wireshark show that the active entity's SESS_INIT received matches intended node ID and the SESS_INT is rejected if the two do not match.

TCPCL-020 Transfer MRU

When the Transfer MRU provided during negotiation is unacceptable, the entity shall terminate the session with a reason code of "Contact Failure”.

Status:	Vetted
Source:	RFC 9174 Section 4.7
Rationale:	Reason codes enable network statistics. These statistics can be used to understand how the network is operating.
Verification Method:	Inspection
Verification Approach:	This requirement shall be considered verified when a code inspection shows a termination and reason code of "Contact Failure" happens when the Transfer MRU is unacceptable.

TCPCL-021 Segment MRU Contact Failure

When the Segment MRU provided during negotiation is unacceptable, the entity shall terminate the session with a reason code of "Contact Failure".

Status:	Vetted
Source:	RFC 9174 Section 4.7
Rationale:	Reason codes enable network statistics. These statistics can be used to understand how the network is operating.
Verification Method:	Inspection
Verification Approach:	This requirement shall be considered verified when a code inspection shows a termination and reason code of "Contact Failure" happens when the Segment MRU is unacceptable.

TCPCL-022 Session Keep-alive Interval

When the Session keep-alive Interval provided during negotiation is unacceptable, the entity shall terminate the session with a reason code of "Contact Failure".

Status:	Draft
Source:	RFC 9174 Section 4.7
Rationale:	Nodes require proper keep-alive intervals and timeouts, so nodes are not "hung up".
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows a contact header being sent that has a bogus keep-alive interval and the receiving entity terminates the session with a reason code of "Contact Failure".

TCPCL-023 Session Extension Items Contact Failure

When a TCPCL entity receives a Session Extension Items Item Flags with an unknown Item Type and the CRITICAL flag is 1, the entity shall refuse the TCPCL session with a SESS_TERM reason code of "Contact Failure".

Status:	Vetted
Source:	RFC 9174 Section 4.8
Rationale:	Reason codes enable network statistics. These statistics can be used to understand how the network is operating.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test shows a TCPCL entity receiving a Session Extension Items Item Flags with an unknown Item Type and the CRITICAL flag = 1 and the entity refusing the TCPCL session with a SESS_TERM reason code of "Contact Failure".

TCPCL-024 Session Extension Items Encoding

Session Extension Items shall use Type-Length-Value (TLV) containers encoding.

Status:	Vetted
Source:	RFC 9174 Section 4.8
Rationale:	This is a requirement from RFC 9174 section 4.8. The fields of Session Extension Items are: item flags, item type, item length, and item value.
Verification Method:	Inspection
Verification Approach:	This requirement shall be considered verified when a code inspection shows Session Extension Items encoded as TLV containers.

TCPCL-025 Keep-alive

Nodes shall send a keep-alive message when no message transmission reception happens during the negotiated interval.

Status:	Vetted
Source:	RFC 9174 Section 5.1.1
Rationale:	Timeouts avoid an idle network. Timeouts ensure the network will not be 'hung up' in any particular node. Keep-alive messages are the network's last chance before a timeout occurs.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test shows an entity sending a keep-alive message when no message transmission reception happens during the negotiated interval between entities.

TCPCL-026 SESS_TERM Idle Timeout

The entity shall terminate the session by transmitting a SESS_TERM message with a reason code of "Idle timeout" when no message transmission reception happens during the negotiated interval.

Status:	Vetted
Source:	RFC 9174 Section 5.1.1
Rationale:	Messages can be keep-alive or other. Reason codes enable network statistics. These statistics can be used to understand how the network is operating.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows an entity terminating the TCP connection from no message transmission reception during the negotiated interval with a reason code of "Idle timeout".

TCPCL-027 Message Unsupported

When a TCPCL entity receives a known message type but is inappropriate for the negotiated session parameters, the entity shall send a MSG_REJECT message with a reason code of "Message Unsupported".

Status:	Vetted
Source:	RFC 9174 Section 5.1.2
Rationale:	This requirement is a catch statement. An inappropriate message can be due to an incorrectly negotiated session extension. Reason codes enable network statistics. These statistics can be used to understand how the network is operating.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows an entity sending a MSG_REJECT message from a wrong message type for the negotiated session parameters with a reason code of "Message Unsupported".

TCPCL-028 Message Unexpected

When a TCPCL entity receives a message that is inappropriate for the current session state, the entity shall send a MSG_REJECT message with a reason code of "Message Unexpected".

Status:	Draft
Source:	RFC 9174 Section 5.1.2
Rationale:	Inappropriate for the current session state can mean a SESS_INIT after the session has already been established or a XFER_ACK message with an unknown Transfer ID.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows an entity sending a MSG_REJECT message with a reason code of "Message Unsupported" when it was sent a known message type that did not make sense for the negotiated session parameters.

TCPCL-029 TCPCL Segment to Segment MRU

A TCPCL segment shall be less than or equal to the receiving entity's Segment MRU.

Status:	Vetted
Source:	RFC 9174 Section 5.2
Rationale:	A receiving entity can set the Segment Maximum Receive Unit (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 using the TCPCL protocol.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows the comparison of a TCPCL segment is less than or equal to the receiving entity's Segment MRU.

TCPCL-030 Single Transfer

A single transfer shall contain a single bundle.

Status:	Vetted
Source:	RFC 9174 Section 5.2
Rationale:	This requirement is imposed on the agent using the TCPCL, rather than on the TCPCL itself. Data must be as one unit since TCPCL is unable to handle fragmentation like the bundle protocol layer.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a Wireshark test shows that a single bundle is always captured in a single TCP transfer.

TCPCL-031 Multiple Bundles

Multiple bundles on a single TCPCL connection shall transmit contiguously.

Status:	Vetted
Source:	RFC 9174 Section 5.2
Rationale:	This is a requirement from RFC 9174 section 5.2. Transmitting multiple bundles contiguously aids the network by not losing a bundle.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a Wireshark test shows that a single TCPCL connection transmits contiguously when there are multiple bundles being sent.

TCPCL-032 Unique Transfer ID

Transfer IDs shall differ between endpoint entities within a single TCPCL session and direction.

Status:	Vetted
Source:	RFC 9174 Section 5.2.1
Rationale:	When transfer IDs are not unique within a single TCPCL session then communications could be accepted at another node. When this happens, the network will lose its validity and integrity.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test of multiple bundles within a single TCPCL session contains different transfer IDs between endpoint entities.

TCPCL-033 Reserved Message Header Flag Set

The sender shall set the reserved message header flag bits to 0.

Status:	Vetted
Source:	RFC 9174 Section 5.2.2
Rationale:	Reserved message header flag bits are found within the Contact Header.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows the sender's reserved message header flag bits are zero.

TCPCL-034 Transfer Extension Items Length

The Transfer Extension Items Length and Transfer Extension Items list shall only be present when the START flag is 1 on the message.

Status:	Vetted
Source:	RFC 9174 Section 5.2.2
Rationale:	Transfer Extension Items Length and Transfer Extension Items list: Together, these fields represent protocol extension data for this specification.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows the Transfer Extension Items Length and Transfer Extension Items are present when the START = 1 on the message and a test showing that the Transfer Extension Items Length and Transfer Extension Items list are NOT present when the START = 0.

TCPCL-035 START Flag

The first segment of a transfer shall set the START flag = 1.

Status:	Vetted
Source:	RFC 9174 Section 5.2.2
Rationale:	The flags portion of the message contains two flag values in the two low-order bits, denoted START and END in XFER_SEGMENT flags. These flags are to start and stop a handshake.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test shows the first segment of a transfer contains START = 1 and END = 0.

TCPCL-036 END Flag

The last segment of a transfer shall set the END flag = 1.

Status:	Vetted
Source:	RFC 9174 Section 5.2.2
Rationale:	The flags portion of the message contains two flag values in the two low-order bits, denoted START and END in XFER_SEGMENT flags. These flags are to start and stop a handshake.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test shows the last segment of a transfer contains START = 0 and END = 1.

TCPCL-037 Bundle Transfer

When a transfer of a bundle has commenced, the entity shall only send segments containing sequential portions of that bundle to the END flag = 1 segment.

Status:	Vetted
Source:	RFC 9174 Section 5.2.2
Rationale:	The flags portion of the message contains two flag values in the two low-order bits, denoted START and END in XFER_SEGMENT flags. These flags are to start and stop a handshake.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that interleaving of multiple transfers from the same entity is not possible within a single TCPCL session.

TCPCL-038 Transfer ACK

A receiving TCPCL entity shall send XFER_ACK message(s) in response to receiving processed XFER_SEGMENT message segment(s).

Status:	Vetted
Source:	RFC 9174 Section 5.2.3
Rationale:	The rationale behind these acknowledgments is to enable the transmitting entity to determine how much of the bundle has been received, so that if the session is interrupted, it can perform reactive fragmentation to avoid resending the already transmitted part of the bundle. In addition, there is no explicit flow control on the TCPCL.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test shows a receiving TCPCL entity sending XFER_ACK messages in response to the same amount of receiving processed XFER_SEGMENT message segments.

TCPCL-039 XFER_ACK Flags

The flags portion of the XFER_ACK header shall match the acknowledged corresponding XFER_SEGMENT message.

Status:	Vetted
Source:	RFC 9174 Section 5.2.3
Rationale:	The flags portion include flags that are not decodable to the entity. The rationale behind these acknowledgments is to enable the transmitting entity to determine how much of the bundle has been received, so that if the session is interrupted, it can perform reactive fragmentation to avoid resending the already transmitted part of the bundle. In addition, there is no explicit flow control on the TCPCL.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test shows the flags portion of the XFER_ACK header matches the same number of receiving processed XFER_SEGMENT message segments.

TCPCL-040 Transfer Refuse ID

The transfer sender shall indicate the transfer ID of a refused transfer.

Status:	Vetted
Source:	RFC 9174 Section 5.2.4
Rationale:	By indicating the transfer ID of a refused transfer, the network statics can aid the overall network health including the health of each individual node.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test shows that the transfer sender indicates the transfer ID of a refused transfer that was sent.

TCPCL-041 Complete Transmission

When a sender receives a XFER_REFUSE message, the sender shall complete the transmission of partially sent XFER_SEGMENT message(s).

Status:	Vetted
Source:	RFC 9174 Section 5.2.4
Rationale:	This handshake lets the sender know what was received.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test shows a sender completing its transmission of partially sent XFER_SEGMENT messages after receiving a XFER_REFUSE message.

TCPCL-042 Refused Bundle XFER_REFUSE

Subsequent segments of a refused bundle shall be refused with a XFER_REFUSE message.

Status:	Draft
Source:	RFC 9174 Section 5.2.4
Rationale:	For network management purposes it is important to label all parts of the message so if a segment is received the receiving entity knows what it should do.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that it labels all subsequent segments of a refused bundle with a XFER_REFUSE message.

TCPCL-043 Transfer Extension Items

Transfer Extension Items shall use Type-Length-Value (TLV) containers encoding.

Status:	Vetted
Source:	RFC 9174 Section 5.2.5
Rationale:	Transfer Extension Items are defined in RFC 9174 section 5.2.5. Transfer Extension item fields are: item flags, item type, item length, and item value.
Verification Method:	Inspection
Verification Approach:	This requirement shall be considered verified when a code inspection shows Transfer Extension Items encoded as TLV containers.

TCPCL-044 CRITICAL Flag 1

When a TCPCL entity receives a Transfer Extension Item with an unknown Item Type and the CRITICAL flag is 1, then the entity shall refuse the transfer with a XFER_REFUSE reason code of "Extension Failure".

Status:	Vetted
Source:	RFC 9174 Section 5.2.5
Rationale:	Reason codes enable network statistics. These statistics can be used to understand how the network is operating.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test shows a TCPCL entity receiving a Transfer Extension Item with an unknown Item Type and the CRITICAL flag = 1 and the entity refusing the transfer with a XFER_REFUSE reason code of "Extension Failure".

TCPCL-045 Single Transfer Length Extension Items

At the start of a bundle, a transfer that contains segments shall contain a single Transfer Length Extension Item.

Status:	Vetted
Source:	RFC 9174 Section 5.2.5
Rationale:	This requirement is imposed on the agent using the TCPCL, rather than on the TCPCL itself.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test showing a bundle transfer with multiple segments contains a single Transfer Length Extension Item.

TCPCL-046 IANA Code Point

The Transfer Length Extension shall use the IANA-assigned code point.

Status:	Vetted
Source:	RFC 9174 Section 5.2.5
Rationale:	IANA has created the "Bundle Protocol TCP Convergence-Layer Version 4 Transfer Extension Types" registry and populated it with the contents of the Transfer Length Extension Codes. Values in the range 0x8000-0xFFFF are reserved for Private or Experimental Use, which are not recorded by IANA.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows the Transfer Length Extension value is assigned by the IANA-assigned code point options.

TCPCL-047 Total Length Field

The receiver shall accept a received bundle when the Total Length value matches the length of the bundle data received.

Status:	Vetted
Source:	RFC 9174 Section 5.2.5
Rationale:	The total length mandates what the length is going to be. The authority that the receiver has is to accept it or not.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows a received bundle being accepted based on the Total Length value matching the length of the bundle data received.

TCPCL-048 Actual Total Length

When the actual total length of bundle data received is different from the value indicated by the Total Length value, then the receiver shall:

1. Invalidate the transmitted data.

2. Send a XFER_REFUSE with a reason code of "Not Acceptable".

Status:	Vetted
Source:	RFC 9174 Section 5.2.5
Rationale:	Reason codes enable network statistics. These statistics can be used to understand how the network is operating.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows a receiver invalidating transmitted data and sending a XFER_REFUSE message with a reason code of "Not Acceptable" when the actual total length of bundle data received is different from the value indicated by the Total Length value.

TCPCL-049 Session Termination

To terminate a session, an entity shall

1. Complete transmission of other message(s).

2. Transmit a SESS_TERM message.

Status:	Vetted
Source:	RFC 9174 Section 6.1
Rationale:	Must finish the handshake for both parties to terminate the session.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed showing one entity terminating a session (by SESS_TERM message) and the other entity completing transmission of its message and transmitting its SESS_TERM message.

TCPCL-050 SESS_TERM Message

When initiating a termination, the REPLY flag of a SESS_TERM message shall be set to 0.

Status:	Vetted
Source:	RFC 9174 Section 6.1
Rationale:	Setting the REPLY flag of a SESS_TERM message lets the other node know that its connection wants to have a clean close out.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test shows an entity terminating a session by sending the REPLY = 0 in a SESS_TERM message and a test showing that an entity with REPLY = 1 in a SESS_TERM message does not terminate the session.

TCPCL-051 Acknowledging SESS_TERM

Upon receiving an initial SESS_TERM message in the current session, an entity shall send an acknowledging SESS_TERM message.

Status:	Vetted
Source:	RFC 9174 Section 6.1
Rationale:	The receiver node must acknowledge the sender to complete a handshake.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test shows that a receiving SESS_TERM message entity, sends its acknowledging SESS_TERM message.

TCPCL-052 XFER_REFUSE

While the Ending state attempts a new incoming transfer, the receiving entity shall send a XFER_REFUSE with a reason code of "Session Terminating".

Status:	Vetted
Source:	RFC 9174 Section 6.1
Rationale:	Reason codes enable network statistics. These statistics can be used to understand how the network is operating.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows an entity sending a XFER_REFUSE message from being in an Ending state with a reason code of "Session Terminating".

TCPCL-053 TCP Closed Failed Transfer

When the underlying TCP connection closes during a transmission, the BPA shall indicate the failed transfer.

Status:	Vetted
Source:	RFC 9174 Section 6.1
Rationale:	TCP connections can be closed in either transfer stream.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when HDTN receives an error message that the TCP socket is closed when the underlying TCP connection is closed.

TCPCL-054 Contact Header Fails TCP Closes

When reception of the Contact Header fails, an entity shall close the TCP connection without sending a SESS_TERM message.

Status:	Vetted
Source:	RFC 9174 Section 6.1
Rationale:	Contact Header is sent first to make certain the two nodes are compatible. The SESS_TERM message sets the final settings of the communication. A reception of the contact header can fail by receiving an invalid magic string.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows an entity closing the TCP connection without sending a SESS_TERM message after it received a failed Contact Header.

TCPCL-055 Ending State

While the session is in the Ending state, an entity shall complete the termination transfer procedure for the remainder of the session.

Status:	Vetted
Source:	RFC 9174 Section 6.1
Rationale:	When the handshake/session is in the termination process, the two nodes are focused on a clean close-out. Therefore, no new transmission and receiver messages are able to be handled.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that when a session is in the ending state, no new incoming and outgoing transfers are ignored.

6 STCP REQUIREMENTS

STCP-001 Bundle Reception

Bundles shall be received over the connected TCP socket.

Status:	Vetted
Source:	STCPCLI
Rationale:	STCPCL was implemented for compatibility with ION.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that initializes an STCPCL outduct and confirms a TCP connection is used as the protocol for bundle transfer.

STCP-002 Bundle Reception Length

STCP shall interpret a 32-bit unsigned integer preceding a bundle in network byte order as the length of the bundle.

Status:	Vetted
Source:	STCPCLI
Rationale:	The 32-bit length frames the bundle within a TCP stream.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that sends a bundle to an STCPCL induct and confirms the initial 32-bit integer of the bundle is read as the bundle length and is used to read the correct number of bytes from the TCP stream.

STCP-003 Unidirectional Link

Each STCPCL link shall be unidirectional.

Status:	Vetted
Source:	STCPCL
Rationale:	The next hop node ID is unknown.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when each STCP link has been shown to be unidirectional.

STCP-004 Bundle Transmission

Bundles shall be transmitted over the connected TCP socket.

Status:	Vetted
Source:	STCPCLO
Rationale:	STCP was implemented for compatibility with ION.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when bundles are transmitted over the TCP connection.

STCP-005 Bundle Transmission Length

Bundles transmitted on the connection shall be preceded by a 32−bit unsigned integer in network byte order indicating the length of the bundle.

Status:	Vetted
Source:	STCPCLO
Rationale:	The 32-bit length frames the bundle within a TCP stream.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that receives a bundle from an STCPCL outduct and confirms the value of the 32-bit integer preceding the bundle matches the length of the bundle.

STCP-006 STCPCL Keep-alive

Keep-alive packets shall be indicated by a 32-bit unsigned integer with all bits set to zero.

Status:	Vetted
Source:	STCPCL
Rationale:	Keep-alive packets have been implemented for compatibility with ION STCPCL.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that monitors an STCPCL outduct and confirms the TCP session is maintained by periodic reception of data or 32-bit unsigned integer with all bits set to zero.

STCP-007 STCPCL Keep-alive Interval

Keep-alive packets shall be transmitted at a user specified interval when no data is being transmitted.

Status:	Vetted
Source:	STCPCL
Rationale:	Keep-alive packets have been implemented for compatibility with ION STCPCL.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that monitors an STCPCL outduct and confirms keep-alives are transmitted at the user specified interval.

STCP-008 Maximum Bundle Length

Accepted bundles shall be less than or equal to a user specified maximum length.

Status:	Vetted
Source:	STCPCLI
Rationale:	Very large bundles could consume a large amount of memory.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that sends an oversized bundle to an STCPCL induct and confirms the connection is terminated.

7 UDP REQUIREMENTS

UDPCL-001 BP Over a Datagram CL

A UDP datagram shall contain one bundle.

Status:	Vetted
Source:	UDP RFC 7122 Section 3.2
Rationale:	In order to utilize DTN protocols across the Internet, it is necessary to encapsulate them into a standard Internet Protocol so that they travel easily across the Internet. UDP is a unidirectional protocol with no congestion control. One bundle per datagram simplifies the convergence layer.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that receives a datagram from a UDPCL outduct and confirms the datagram contains exactly one bundle.

UDPCL-002 UDP Keep-alive

UDPCL inducts shall interpret receiving 4 octets of zero bits as a keep-alive.

Status:	Draft
Source:	UDP RFC 7122 Section 3.2
Rationale:	UDPCL inducts support receiving keep-alives for compatibility with other implementations.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when inducts accept 4 octets of zero bits as keep-alives, does not log an error, and does not attempt to deliver them as data.

UDPCL-003 Bundle Fragmentation

Bundles shall be fragmented into UDP datagrams based on a configured MTU.

Status:	Draft
Source:	UDP RFC 7122 Section 3.2
Rationale:	Bundles that are too large for the path MTU shall be fragmented and reassembled to prevent IP fragmentation.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when bundles are fragmented into UDP datagrams that match the configured MTU size.

UDPCL-004 Rate Limit

UDP outduct transmission rates shall be limited by a user specific rate limit.

Status:	Vetted
Source:	UDP RFC 7122 Section 3.6
Rationale:	UDP packets will be dropped if they exceed the expected link rate.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that receives data from a UDPCL outduct and confirms the received data rate does not exceed the configured rate.

UDPCL-005 Port Configuration Listening

UDP induct shall bind to an user provided port.

Status:	Vetted
Source:	UDP RFC 7122 Section 3.2.2
Rationale:	UDP connections are to be established via a known port.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that connects successfully to an UDPCL induct utilizing the expected port.

UDPCL-006 Port Configuration Transmission

UDP outduct shall connect to an user provided port.

Status:	Vetted
Source:	UDP RFC 7122 Section 3.2.2
Rationale:	UDP connections are to be established via a known port.
Verification Method:	Test
Verification Approach:	This requirement shall be considered verified when a test is performed that receives a connection from an UDPCL outduct utilizing the expected port.

8 LTP REQUIREMENTS

LTP-001 Session ID

A session ID shall consist of the engine ID of the sender and a session number randomly generated by the sender.

Status:	Vetted
Source:	RFC 5326 Section 3.1
Rationale:	This is the definition of the session ID.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that a series of session IDs consists of the engine ID of the sender and a session number randomly generated by the sender.

LTP-002 Unique Session ID

Every session initiated by an LTP engine shall be uniquely identified by a session ID.

Status:	Vetted
Source:	RFC 5326 Section 3.1
Rationale:	Each session needs a unique ID to distinguish it from other sessions.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that a series of bundles transmitted from an LTP outduct contains unique session IDs.

LTP-003 Incrementing Checkpoint Serial Numbers

Any subsequent checkpoints issued by the sender shall have the serial number value found by incrementing the prior checkpoint serial number by 1.

Status:	Vetted
Source:	RFC 5326 Section 3.2.1
Rationale:	The checkpoint serial number uniquely identifies the checkpoint among all checkpoints issued by the block sender in a session.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that a series of checkpoint serial numbers increment by one for each subsequent checkpoint.

LTP-004 Retransmitted Checkpoints

When a checkpoint segment is retransmitted, its serial number shall be the same as when it was originally transmitted.

Status:	Vetted
Source:	RFC 5326 Section 3.2.1
Rationale:	Retransmitted serial numbers should not be changed from the original serial number since it is related to the same checkpoint.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the serial number for a retransmitted checkpoint is the same as the serial number of the original checkpoint.

LTP-005 Report Serial Number

When the checkpoint is queued for transmission in response to the reception of an RS, then its value shall be the report serial number value of the RS that caused the data segment to be queued for transmission.

Status:	Vetted
Source:	RFC 5326 Section 3.2.1
Rationale:	This is required according to section 6.13 of RFC 5326.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the report serial number of a received RS matches the RS that caused the data segment to be queued for transmission.

LTP-006 Report Segment

Any subsequent Report Segment issued by the receiver shall have the serial number value found by incrementing the last report serial number by 1.

Status:	Vetted
Source:	RFC 5326 Section 3.2.2
Rationale:	Need to implement per RFC 5326 section 3.2.2 and helps ensure interoperability with other DTNs.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that a series of Report Segments have a report serial number that increments by 1.

LTP-007 Retransmitted Report Segment

When a Report Segment is retransmitted, its serial number shall be the same as when it was originally transmitted.

Status:	Vetted
Source:	RFC 5326 Section 3.2.2
Rationale:	Need to implement per RFC 5326 section 3.2.2 and helps ensure interoperability with other DTNs.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that a retransmitted Report Segment contains its original serial number.

LTP-008 Report Serial Number Greater Than Zero

The report serial number shall be greater than zero.

Status:	Vetted
Source:	RFC 5326 Section 3.2.2
Rationale:	Need to implement per RFC 5326 section 3.2.2 and helps ensure interoperability with other DTNs.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that the report serial number disallows zero.

LTP-009 Checkpoint Serial Number

The value of the checkpoint serial number shall be zero if the report segment is not a response to reception of a checkpoint.

Status:	Draft
Source:	RFC 5326 Section 3.2.2
Rationale:	Need to implement per RFC 5326 section 3.2.2 and it's used to differentiate from a response to reception of a checkpoint.
Verification Method:	Needs further discussion.
Verification Approach:	Draft requirement. Needs further discussion.

LTP-010 Checkpoint Serial Number RS

The value of the checkpoint serial number shall be the checkpoint serial number of the checkpoint that caused the RS to be issued if the report segment is a response to reception of a checkpoint.

Status:	Draft
Source:	RFC 5326 Section 3.2.2
Rationale:	Need to implement per RFC 5326 section 3.2.2 and it's used to differentiate from a response to reception of a checkpoint.
Verification Method:	Needs further discussion.
Verification Approach:	Draft requirement. Needs further discussion.

LTP-011 Reception Claims Length Limits

The sum of an LTP Reception Claim's length and offset shall not exceed the difference between the upper and lower bounds of the report segment.

Status:	Vetted
Source:	RFC 5326 Section 3.2.2
Rationale:	Need to implement per RFC 5326 section 3.2.2.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the sum of a LTP Reception Claim's length and offset is between the upper and lower bounds of the report segment.

LTP-012 Reception Claims Offset

An LTP Reception Claim's offset shall be greater than the sum of the offset and the length of the prior claim.

Status:	Vetted
Source:	RFC 5326 Section 3.2.2
Rationale:	Need to implement per RFC 5326 section 3.2.2.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that a LTP Reception Claim's offset is greater than the sum of the offset and the length of the prior claim.

LTP-013 Reception Claims Upper Bound

An LTP Reception Claim's upper bound shall be greater than or equal to the sum of the offset, length and lower bound.

Status:	Vetted
Source:	RFC 5326 Section 3.2.2
Rationale:	Need to implement per RFC 5326 section 3.2.2.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that an LTP Reception Claim's upper bound is greater than or equal to the sum of the offset, length and lower bound.

LTP-014 Session Management Segments

An LTP Session Management Segment shall consist of cancel segment or cancel acknowledgement.

Status:	Vetted
Source:	RFC 5326 Section 3.2.4
Rationale:	Need to implement per RFC 5326 section 3.2.4.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that an LTP Session Management Segment contains either a cancel segment or cancel acknowledgement.

LTP-015 End of Block

The last data segment in a block shall be marked as the EOB (end of block).

Status:	Vetted
Source:	RFC 5326 Section 4.1
Rationale:	Need to implement per RFC 5326 section 4.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that last data segment in a block is marked as the EOB (end of block).

LTP-016 Data Segment

An LTP data segment shall only contain either red-part data, green-part data, or both.

Status:	Vetted
Source:	RFC 5326 Section 4.1
Rationale:	Need to implement per RFC 5326 section 4.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that a LTP data segment contains either red-part, green-part, or both.

LTP-017 End of Red-Part

The Last Data Segment for a Red-part data shall be marked as the EORP (end of red-part) segment.

Status:	Vetted
Source:	RFC 5326 Section 4.1
Rationale:	Need to implement per RFC 5326 section 4.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that last data segment for a red-part data is marked as the EORP (end of red-part) segment.

LTP-018 Maximum Transmission Unit

Data shall be subdivided into data segments within a user specified maximum transmission unit size.

Status:	Vetted
Source:	RFC 5326 Section 4.1
Rationale:	Need to implement per RFC 5326 section 4.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that data segments are within the specified maximum transmission unit size.

LTP-019 Requirements from the Operating Environment

LTP shall be run directly over a data-link layer protocol.

Status:	Vetted
Source:	RFC 5326 Section 5
Rationale:	LTP is meant to provide additional reliability on top of an existing data link layer.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows the LTP implementation utilizes a lower-level data-link layer protocol.

LTP-020 LTP Link Status

The LTP Engine shall detect the status of an LTP destination.

Status:	Vetted
Source:	RFC 5326 Section 5
Rationale:	Need to implement per RFC 5326 section 5. LTP Engine needs to detect if the link was brought up or turn down.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the LTP Engine detects if the link to the destination is active.

LTP-021 One-way Light Time

The LTP Engine shall read the current distance from the configuration file.

Status:	Vetted
Source:	RFC 5326 Section 5
Rationale:	Need to implement per RFC 5326 section 5. This is used to calculate timeout intervals.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the one-way light time parameter is read correctly from the configuration file.

LTP-022 Local Data-link Layer Protocols

The content of each local data-link layer protocol frame shall contain an integer number of LTP segments.

Status:	Vetted
Source:	RFC 5326 Section 5
Rationale:	Need to implement per RFC 5326 section 5. Fractions of LTP segments would lack the needed context and could become invalid if received out of order.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the content of each local data-link layer protocol frame is an integral number of LTP segments.

LTP-023 Invalid Segments

The LTP Engine shall discard invalid segments.

Status:	Vetted
Source:	RFC 5326 Section 6
Rationale:	Need to implement per RFC 5326 section 6. LTP segments that do not conform to the specification are discarded.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the LTP Engine discards invalid segments.

LTP-024 UNREACH Reason Code

The LTP Engine shall send a CR with reason-code UNREACH if the invalid data segment contains red-part data.

Status:	Vetted
Source:	RFC 5326 Section 6
Rationale:	Need to implement per RFC 5326 section 6.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the LTP Engine sends a CR with reason-code UNREACH when an invalid data segment contains red-part data.

LTP-025 Retransmit Checkpoint

The expiration of a countdown timer associated with a CP segment shall invoke the Cancel Session procedure for the session associated with this segment.

Status:	Vetted
Source:	RFC 5326 Section 6.7
Rationale:	Need to implement per RFC 5326 section 6.7. As countdowns expire, they must end the corresponding session via the cancel session procedure and retransmission.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that expiration of a countdown timer associated with a CP segment triggers the Cancel Session procedure.

LTP-026 Retransmit RS

When the number of times any affected RS segment has been queued for transmission exceeds the report retransmission limit established for the local LTP engine, then the "Cancel Session" procedure shall be invoked.

Status:	Vetted
Source:	RFC 5326 Section 6.8
Rationale:	Need to implement per RFC 5326 section 6.8. Need to free up resources if retransmission limit is exceeded by cancelling session.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the "cancel session" procedure is invoked when the retransmission attempts of RS segment(s) exceeds the report retransmission limit.

LTP-027 Signify Red-Part Reception

Upon the arrival of a CP segment when the EORP for this session has been received and all data in the red-part of the block being transmitted in this session have been received, the LTP engine shall send a red-part reception notice to the specified client service.

Status:	Vetted
Source:	RFC 5326 Section 6.9
Rationale:	Need to implement per RFC 5326 section 6.9. The LTP engine needs to notify the sender when all of red-part has been received.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the LTP engine sends a red-part reception notice to the specified client service upon the arrival of a CP segment at the end-of-red part and all data in the red-part of the block being transmitted in this session have been received.

LTP-028 Signify Green-Part Segment Arrival

Upon the arrival of a data segment whose content is a portion of the green-part of a block, the LTP engine shall send a green-part segment arrival notice to the specified client service.

Status:	Vetted
Source:	RFC 5326 Section 6.10
Rationale:	Need to implement per RFC 5326 section 6.10. The LTP engine needs to notify the sender when any green-part arrives.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that upon the arrival of a data segment whose content is a portion of the green-part of a block, the LTP engine sends a green-part segment arrival notice to the specified client service.

LTP-029 Send Reception Report

When the number of reception problems detected for this session exceeds a limit established for the local LTP engine, then the "Cancel Session" procedure shall be invoked.

Status:	Vetted
Source:	RFC 5326 Section 6.11
Rationale:	Need to implement per RFC 5326 section 6.11. Resources must be freed if the reception problems limit is exceeded by the cancelling session.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the cancel session procedure is invoked when the number of reception problems detected for this session exceeds the limit established for the local LTP engine.

LTP-030 Signify Transmission Completion Notice

A transmission-session completion notice shall be sent to the local client service associated with the session when these conditions have been met:

1. Data in the block is known to have been transmitted.

2. The entire red-part of the block is known to have been successfully received.

Status:	Vetted
Source:	RFC 5326 Section 6.12
Rationale:	Need to implement per RFC 5326 section 6.12. Need to notify sender when entire transmission has been received, and all data is accounted for.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that a transmission-session completion notice is sent to the local client service associated with the session when these conditions have been met: 1. Data in the block is known to have been transmitted. 2. The entire red-part of the block is known to have been successfully received.

LTP-031 Reason Code RLEXC

When the number of transmission problems for this session exceeds a limit established for the local LTP engine, a CS with reason-code RLEXC shall be appended to the transmission queue specified in the transmission request that started this session, and a transmission-session cancellation notice is sent back to the client service that requested the transmission.

Status:	Vetted
Source:	RFC 5326 Section 6.13
Rationale:	There should only be a finite number of retransmission attempts. See section RFC 5326 section 7.5 for the transmission session cancellation notice.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the number of transmission problems for a session exceeds a limit established for the local LTP engine, a CS with reason-code RLEXC shall be appended to the transmission queue specified in the transmission request that started the session, and a transmission-session cancellation notice is sent back to the client service that requested the transmission.

LTP-032 Stop RS Timer

The countdown timer associated with the original RS segment (identified by the report serial number of the RA segment) shall be deleted on the reception of an RA.

Status:	Vetted
Source:	RFC 5326 Section 6.14
Rationale:	Need to implement per RFC 5326 section 6.14. Resources should be freed by ending countdown timers associated with RS segments.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that countdown timer associated with the original RS segment (identified by the report serial number of the RA segment) is deleted on the reception of an RA.

LTP-033 CS Start Cancel Timer

Upon the arrival of a link state cue indicating the de-queuing (for transmission) of a CS segment, a countdown timer for the expected arrival time of the CAS segment shall be started.

Status:	Vetted
Source:	RFC 5326 Section 6.15
Rationale:	Need to implement per RFC 5326 section 6.15. Timers are needed to diagnose connection conditions.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that upon the arrival of a link state cue indicating the de-queuing (for transmission) of a CS segment, a countdown timer for the expected arrival time of the CAS segment is started.

LTP-034 CR Start Cancel Timer

Upon the arrival of a link state cue indicating the de-queuing (for transmission) of a CR segment, a countdown timer for the expected arrival time of the CAR segment shall be started.

Status:	Vetted
Source:	RFC 5326 Section 6.15
Rationale:	Need to implement per RFC 5326 section 6.15. Timers are needed to diagnose connection conditions.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that upon the arrival of a link state cue indicating the de-queuing (for transmission) of a CR segment, a countdown timer for the expected arrival time of the CAR segment is started.

LTP-035 CS Acknowledge Cancellation

When a CS segment has a transmission queue-set bound for the sender, a CAS (cancel acknowledgment to block sender) segment shall be appended to the queue of internal operations traffic bound for the sender.

Status:	Vetted
Source:	RFC 5326 Section 6.17
Rationale:	Need to implement per RFC 5326 section 6.17. The LTP engine needs to acknowledge when a cancel session segment is received.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that if a CS segment has a transmission queue-set bound for the sender, a CAS (cancel acknowledgment to block sender) segment is appended to the queue of internal operations traffic bound for the sender.

LTP-036 CR Acknowledge Cancellation

When a CR segment has a transmission queue-set bound for the sender, a CAR (cancel acknowledgment to block receiver) segment shall be appended to the queue of internal operations traffic bound for the receiver.

Status:	Vetted
Source:	RFC 5326 Section 6.17
Rationale:	Need to implement per RFC 5326 section 6.17. The LTP engine needs to acknowledge when a cancel session segment is received.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that if a CR segment has a transmission queue-set bound for the sender, a CAR (cancel acknowledgment to block receiver) segment is appended to the queue of internal operations traffic bound for the receiver.

LTP-037 CAS Stop Cancel Timer

Upon reception of a CAS segment, the timer associated with the CS segment shall be deleted.

Status:	Vetted
Source:	RFC 5326 Section 6.18
Rationale:	Need to implement per RFC 5326 section 6.18. Resources must be freed by ending countdown timers associated with CS segments.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that upon reception of a CAS segment, the timer associated with the CS segment is deleted.

LTP-038 CAR Stop Cancel Timer

Upon reception of a CAR segment, the timer associated with the CR segment shall be deleted.

Status:	Vetted
Source:	RFC 5326 Section 6.18
Rationale:	Need to implement per RFC 5326 section 6.18. Resources must be freed by ending countdown timers associated with CR segments.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that upon reception of a CAR segment, the timer associated with the CR segment is deleted.

LTP-039 Cancel Session

When a session is canceled, the LTP engine shall delete all queued segments from outbound traffic queues.

Status:	Vetted
Source:	RFC 5326 Section 6.19
Rationale:	Need to implement per RFC 5326 section 6.19. Resources associated with a cancelled session must be freed.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that the LTP engine deletes all queued segments from outbound traffic queues when a session has been canceled.

LTP-040 Countdown Timers

When a session is cancelled, the LTP engine shall delete all countdown timers currently associated with the session.

Status:	Vetted
Source:	RFC 5326 Section 6.19
Rationale:	Need to implement per RFC 5326 section 6.19. Resources associated with a cancelled session must be freed.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that the LTP engine deletes all countdown timers currently associated with a session that has been canceled.

LTP-041 Cancel Session Buffer

When the local LTP engine is the sender, then the remaining data retransmission buffer space allocated to a canceled session shall be released.

Status:	Vetted
Source:	RFC 5326 Section 6.19
Rationale:	Need to implement per RFC 5326 section 6.19. Resources associated with a cancelled session must be freed.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that if the local LTP engine is the sender, then the remaining data retransmission buffer space allocated to a canceled session is released.

LTP-042 Close Session

Remaining countdown timers associated with a closed session shall be deleted.

Status:	Vetted
Source:	RFC 5326 Section 6.20.
Rationale:	Need to implement per RFC 5326 section 6.20. Resources associated with a closed session must be freed.
Verification Method:	Inspection
Verification Approach:	This requirement will be considered verified when a code inspection shows that the remaining countdown timers associated with a closed session are deleted.

LTP-043 Handling Miscolored Segments

When miscolored data blocks are received, the LTP engine shall discard them.

Status:	Vetted
Source:	RFC 5326 Section 6.21
Rationale:	Need to implement per RFC 5326 section 6.21. Need to remove miscolored data blocks from system. This procedure is triggered by the arrival of either (a) a red-part data segment whose block offset begins at an offset higher than the block offset of any green-part data segment previously received for the same session or (b) a green-part data segment whose block offset is lower than the block offset of any red-part data segment previously received for the same session. The arrival of a segment matching either of the above checks is a violation of the protocol requirement of having all red-part data as the block prefix and all green-part data as the block suffix.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that miscolored data blocks are discarded.

LTP-044 Cancel Session for Miscolored Data

When miscolored data blocks are received, the local LTP engine shall cancel the session.

Status:	Vetted
Source:	RFC 5326 Section 6.21
Rationale:	Need to implement per RFC 5326 section 6.21. The LTP engine should prevent further miscolored blocks from entering system.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the LTP session is canceled when miscolored data has been received.

LTP-045 UDP Port Number for LTP

The UDP port number shall be user configurable.

Status:	Vetted
Source:	RFC 5326 Section 10.1
Rationale:	Need to implement per RFC 5326 section 10.1.
Verification Method:	Test
Verification Approach:	This requirement will be considered verified when a test shows that the LTP port is configured based on the configuration file.

APPENDIX A.—ABBREVIATIONS

Abbreviation	Definition
BP v	Bundle Protocol Version
BP	Bundle Protocol
URI	Uniform Resource Identifier
ipn	Interplanetary Network
ASCII	American Standard Code for Information Interchange
CCSDS	Consultative Committee for Space Data Systems
RFC	Request for Comments
BPGen	Bundle Protocol Generator
BPSendFile	Bundle Protocol Send File
HDTN	High-rate DTN
CBHE	Compressed Bundle Header Encoding
ID	Identification
DTN	Delay Tolerant Networking
LTP	Licklider Transmission Protocol
CL	Convergence Layer
SANA	Space Assigned Numbers Authority
UDP	User Datagram Protocol
SDNV	Self-Delimiting Numeric Value
MSB	Most significant bit
MIB	Management Information Base
EID	Endpoint Identifier
CBOR	Concise Binary Object Representation
CRC	Cyclic redundancy check
ADU	Application Data Unit
SSP	Scheme Specific Par
IANA	Internet Assigned Numbers Authority
BPSec	Bundle Protocol Security
BPAs	Bundle Protocol Agents
Sops	Security Operations
BCB	Block Confidentiality Block
BIB	Block Integrity Block
ASB	Abstract Security Block
AEAD	Authenticated Encryption with Associated Data
SABR	Schedule Aware Bundle Routing
CGR	Contact Graph Routing
CMR	Contact Multigraph Routing
TCPCL	Transmission Control Protocol convergence layer
TCP	Transmission Control Protocol
FIN	Finish
CAN_TLS	Contact header flag to enable Transport Layer Security
TLS	Transport layer security
UTF	Unicode Transformation Format

Abbreviation	Definition
SSL	Secure Sockets Layer
CA	Certificate Authority
SESS_INIT	Session Initialization message
MRU	Maximum Receive Unit
SESS_TERM	Session Termination message
MSG_REJECT	Message reject
XFER_ACK	Transfer acknowledge
XFER_SEGMENT	Transfer segment
XFER_REFUSE	Transfer refuse
TLV	Type Length Value
STCP	Simple Transmission Control Protocol
ION	Interplanetary Overlay Network
STCPCL	STCP Convergence Layer
UDPCL	User Datagram Protocol convergence layer
MTU	Maximum transmission unit
RS	Report Segment
EOB	End of block
EORP	End of red-part
CR	Cancel segment from block receiver
CP	Checkpoint
RLEXC	Retransmission Limit Exceeded
CS	Cancel segment
RA	Report-acknowledgment
CAS	Cancel-acknowledgment segment
CAR	Cancel acknowledgement to block receiver

APPENDIX B.—REVISION HISTORY

Revision	Effective Date	Description	Author
Draft	03/22/2024	Initial Requirement Analysis	Rachel Dudukovich