NASA/TM-20240015587

HDTN Software Data Dictionary

Eric Brace
HX5, LLC, Brook Park, Ohio

José Lombay-González, Brian Tomko, Nadia Kortas, Stephanie Booth, Rachel Dudukovich, Ethan Schweinsberg, Amber Waid, and John Nowakowski
Glenn Research Center, Cleveland, Ohio

Wade A. Smith
Bastion Technologies Inc., Houston, Texas

January 2025

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.

- CONFERENCE PUBLICATION.
  Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored 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-20240015587

HDTN Software Data Dictionary

Eric Brace
HX5, LLC, Brook Park, Ohio

José Lombay-González, Brian Tomko, Nadia Kortas, Stephanie Booth, Rachel Dudukovich, Ethan Schweinsberg, Amber Waid, and John Nowakowski
Glenn Research Center, Cleveland, Ohio

Wade A. Smith
Bastion Technologies Inc., Houston, Texas

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

January 2025

Level of Review: This material has been technically reviewed by technical management.

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

SOMD SCaN is developing new communications technologies to increase the amount of science data returned on future space missions. The 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 Software Data Dictionary (DD) describes the structure and content of HDTN data elements.

HDTN Software Data Dictionary

Eric Brace
HX5, LLC
Brook Park, Ohio 44142

José Lombay-González, Brian Tomko, Nadia Kortas, Stephanie Booth, Rachel Dudukovich, Ethan Schweinsberg, Amber Waid, and John Nowakowski
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Wade A. Smith
Bastion Technologies Inc.
Houston, Texas 77058

1.0 Introduction

This Data Dictionary (DD) describes defined variables, data structures, inputs, outputs, commands, file system structures, telemetry, data files, and configuration files.

1.1 Purpose

The DD serves the following purposes:

- Enable uniform communication among team members on the HDTN project.
- Provide key information needed for the implementation, testing, and maintenance of the HDTN software product.
- Allow developers, maintainers, and analysts to access information about the tables, fields, procedures, processes, and other information used by HDTN software.

1.2 System Overview

An overview of the HDTN software project is documented in HDTN-PLAN-004 High-Rate Delay Tolerant Networking Software Development and Management Plan (SDMP).

1.3 Document Overview

The contents of this DD are based on the SWE–58 requirement of NASA Procedural Requirement (NPR) 7150.2 NASA Software Engineering Requirements. NPR 7150.2 requires projects to develop, record, and maintain software design.

The format and content of this DD are based on GRC-SW-TPLT-DD Data Dictionary Template maintained by the GRC Software Engineering Process Group (SEPG) and the HDTN Document Template provided by HDTN Configuration Management (CM) with guidance from the NASA Software Engineering Handbook (SWEHB).

In some cases, there may be gaps in project/system requirements or input resources to this document. These gaps are captured within this document as “to be determined” (TBD) or “to be resolved” (TBR). A list of all TBD and TBR references in this document are contained in Appendix B TBD/TBR List.

Section 3.0 Constant Data defines constant values used by the software.

Section 4.0 Enumeration Data defines enumerated integral constant values used throughout the software.

Section 5.0 Data Types defines custom data types used by the software.

Section 6.0 Data Structures defines data structures used throughout the software.

Section 7.0 Input/Output Data defines input and output formats.

Section 8.0 Application Programming Interface Data defines the purpose, function, format, parameters, and any restrictions for onboard, ground, and test specific commands processed by the software.

Section 9.0 File System Structure defines the structure of the file system used by HDTN software.

Section 10.0 Telemetry Data defines formats and units of measure of data collected from remote sources for transmission to a central location.

Section 11.0 Recorded Data defines data collected and recorded to a storage system.

Section 12.0 Configuration Data defines the format and parameters of data necessary to configure the software.

1.3.1 Data Definitions

Data consumed and produced by HDTN software is defined by the following properties:

- Mnemonic – unique data element identification label used in software implementation
- Description – non-technical name of the data element
- Source – entities that set/send the data element
- Destination – entities that use/receive the data element
- Type – data type; defines the values the element can take and operations that can be performed on it
- Init – value of the element at initialization (if applicable)
    - Values that are not explicitly initialized at the time of declaration are marked with an asterisk (‘*’).
- Min – the minimum acceptable value for the element’s defined range (if applicable)
- Max – the maximum acceptable value for the element’s defined range (if applicable)
- Units – units of measurement represented by the data element

2.0 Referenced Documents

Table 2-1: Reference Documents lists the number and title of each document referenced in this plan.

Table 2-1: Reference Documents

Document Number	Revision	Title	Effective Date
GRC-SW-TPLT-DD	-	Data Dictionary Template	10/31/2012
HDTN-PLAN-004	-	High-Rate Delay Tolerant Networking (HDTN) Software Development and Management Plan (SDMP)	01/17/2024
HDTN-TPLT-023	A	HDTN Document Template	04/19/2024
HDTN-USER-035	base	High-Rate Delay Tolerant Networking (HDTN) User Guide	***TBD-DD001***
NPR 7150.2	D	NASA Software Engineering Requirements	03/08/2022
RFC 6256	-	Using Self-Delimiting Numeric Values in Protocols	05/2011

3.0 Constant Data

The following section defines constant values used in HDTN software.

3.1 Bundle Protocol Version 6 Custody Transfer Constants

Table 3-1: BPv6 Custody Transfer Constants defines the constants specific to Bundle Protocol version 6 (BPv6) Custody Transfer implementation.

Table 3-1: BPv6 Custody Transfer Constants

Mnemonic	Value	Description	Type	Units
NUM_ACS_STATUS_INDICES	BPv6 ACS Status Reason Indices NUM_INDICES	Number of BPv6 ACS status indices	uint	N/A

3.2 Bundle Storage Configuration Constants

Table 3-2: Bundle Storage Configuration Constants defines constants specific to Bundle Storage Configuration implementation.

Table 3-2: Bundle Storage Configuration Constants

Mnemonic	Value	Description	Type	Units
SEGMENT_SIZE	(4096 * STORAGE_SEGMENT_SIZE_MULTIPLE_OF_4KB)	Segment size	macro definition	bytes
SEGMENT_RESERVED_SPACE	28	Segment reserved space	macro definition	bytes
BUNDLE_STORAGE_PER_SEGMENT_SIZE	SEGMENT_SIZE - SEGMENT_RESERVED_SPACE	Bundle storage per segment size	macro definition	bytes
READ_CACHE_NUM_SEGMENTS_PER_SESSION	50	Number of segments to read from cache per session	macro definition	N/A
NUMBER_OF_PRIORITIES	3	Number of priorities	macro definition	N/A
MAX_NUM_STORAGE_THREADS	10	Maximum number of storage threads	macro definition	N/A
CIRCULAR_INDEX_BUFFER_SIZE	30	Circular index buffer size	macro definition	N/A

Definition of the following Bundle Storage Configuration constants is dependent on definition of the STORAGE_SEGMENT_ID_SIZE_BITS symbolic constant.

Table 3-3: Bundle Storage Configuration Constants (STORAGE_SEGMENT_ID_SIZE_BITS = 32) defines the constants specific to Bundle Storage Configuration implementation when STORAGE_SEGMENT_ID_SIZE_BITS is 32.

Table 3-3: Bundle Storage Configuration Constants (STORAGE_SEGMENT_ID_SIZE_BITS = 32)

Mnemonic	Value	Description	Type	Units
SEGMENT_ID_FULL	UINT32_MAX	Maximum number of memory manager tree entries	macro definition	N/A
MAX_MEMORY_MANAGER_ SEGMENTS	UINT32_MAX - 3	Maximum number of memory manager segments	macro definition	N/A

Table 3-4: Bundle Storage Configuration Constants (STORAGE_SEGMENT_ID_SIZE_BITS = 64) defines the constants specific to Bundle Storage Configuration implementation when STORAGE_SEGMENT_ID_SIZE_BITS is 64.

Table 3-4: Bundle Storage Configuration Constants (STORAGE_SEGMENT_ID_SIZE_BITS = 64)

Mnemonic	Value	Description	Type	Units
SEGMENT_ID_FULL	UINT64_MAX	Maximum number of memory manager tree entries	macro definition	N/A
MAX_MEMORY_MANAGER_SEGMENTS	UINT64_MAX - 3	Maximum number of memory manager segments	macro definition	N/A

Definition of the following Bundle Storage Configuration constants is dependent on definition of the _MSC_VER symbolic constant.

Table 3-5: Bundle Storage Configuration Constants (_MSC_VER Defined) defines the constants specific to Bundle Storage Configuration implementation when _MSC_VER is defined.

Table 3-5: Bundle Storage Configuration Constants (_MSC_VER Defined)

Mnemonic	Value	Description	Type	Units
NUM_SEGMENTS_PER_TEST	100,000	Number of segments per test	macro definition	N/A

Table 3-6: Bundle Storage Configuration Constants (_MSC_VER Not Defined) defines the constants specific to Bundle Storage Configuration implementation when _MSC_VER is not defined.

Table 3-6: Bundle Storage Configuration Constants (_MSC_VER Not Defined)

Mnemonic	Value	Description	Type	Units
FILE_SIZE	1,024,000,000	File Size	ULL macro definition	bytes
NUM_SEGMENTS_PER_TEST	100,000	Number of segments per test	macro definition	N/A

3.3 Consultative Committee for Space Data Systems Encapsulation Constants

Table 3-7: CCSDS Encapsulation Constants defines the constants specific to Consultative Committee for Space Data Systems (CCSDS) Encapsulation implementation.

Table 3-7: CCSDS Encapsulation Constants

Mnemonic	Value	Description	Type	Units
CCSDS_ENCAP_PACKET_VERSION_NUMBER	7	CCSDS encapsulation packet version number	macro definition	N/A
SANA_IDLE_ENCAP_PROTOCOL_ID	0	Space Assigned Numbers Authority (SANA) idle encapsulation protocol ID	macro definition	N/A
SANA_LTP_ENCAP_PROTOCOL_ID	1	SANA LTP encapsulation protocol ID	macro definition	N/A
SANA_BP_ENCAP_PROTOCOL_ID	4	SANA BP encapsulation protocol ID	macro definition	N/A
CCSDS_ENCAP_USER_DEFINED_FIELD	0	CCSDS encapsulation user defined field	macro definition	N/A
CCSDS_ENCAP_PROTOCOL_ID_EXT	0	CCSDS encapsulation protocol ID extension	macro definition	N/A
CCSDS_ENCAP_DEFINED_FIELD	0	CCSDS encapsulation defined field	macro definition	N/A

3.4 Contact Graph Routing Constants

Table 3-8: CGR Constants defines the constants specific to Contact Graph Routing (CGR) implementation.

Table 3-8: CGR Constants

Mnemonic	Value	Description	Type	Units
MAX_TIME_T	Maximum value of standard calendar time data type	Max calendar time	time	sec

3.5 Delay Tolerant Networking Real-Time Protocol Constants

Table 3-9: DTN RTP Constants defines the constants specific to DTN Real-Time Protocol (RTP) implementation.

Table 3-9: DTN RTP Constants

Mnemonic	Value	Description	Type	Units
USE_INCOMING_SEQ	TRUE	Use incoming sequence	macro definition	N/A
USE_OUTGOING_SEQ	FALSE	Use outgoing sequence	macro definition	N/A

3.6 Directory Monitor Implementation Constants

Table 3-10: Directory Monitor Implementation Constants defines the constants specific to Directory Monitor Implementation implementation.

Table 3-10: Directory Monitor Implementation Constants

Mnemonic	Value	Description	Type	Units
DIR_MONITOR_USE_DISPATCH_QUEUE	1	Directory monitor use dispatch queue	macro definition	N/A

3.7 GStreamer Application Source Outduct Constants

Table 3-11: GStreamer Application Source Outduct Constants defines the constants specific to GStreamer Application Source Outduct implementation.

Table 3-11: GStreamer Application Source Outduct Constants

Mnemonic	Value	Description	Type	Units
SAMPLE_RATE	9000	Sample rate	macro definition	Hz
DEFAULT_NUM_CIRC_BUFFERS	1,000,000	Default number of circular buffers	macro definition	N/A
GST_HDTN_OUTDUCT_SOCKET_PATH	"/tmp/hdtn_gst_shm_outduct"	GStreamer HDTN outduct socket path	macro definition	N/A
GST_APPSRC_MAX_BYTES_IN_BUFFER	20000000	GStreamer application source maximum bytes in buffer	macro definition	bytes
MAX_NUM_BUFFERS_QUEUE	UINT16_MAX	Maximum number of buffers in queue	macro definition	N/A
MAX_SIZE_BYTES_QUEUE	0	Maximum size in bytes in queue	macro definition	bytes
MAX_SIZE_TIME_QUEUE	0	Maximum size in time in queue	macro definition	nsec
MIN_THRESHHOLD_TIME_QUEUE_NS	500000	Minimum amount of data in the queue to allow reading (in nanoseconds)	macro definition	nsec
RTP_LATENCY_MILLISEC	500	RTP latency (in milliseconds)	macro definition	msec
RTP_MAX_DROPOUT_TIME_MILLISEC	200	RTP maximum time of missing packets tolerated (in milliseconds)	macro definition	msec
RTP_MAX_MISORDER_TIME_MIILISEC	60,000	RTP maximum time of misordered packets tolerated (in milliseconds)	macro definition	msec
RTP_MODE	1	GStreamer RTP jitter buffer mode	macro definition	N/A

3.8 Logger Constants

Table 3-12: Logger Constants defines the constants specific to Logger implementation.

Table 3-12: Logger Constants

Mnemonic	Value	Description	Type	Units
LOG_LEVEL_TRACE	0	Trace log	macro definition	N/A
LOG_LEVEL_DEBUG	1	Debug log	macro definition	N/A
LOG_LEVEL_INFO	2	Info log	macro definition	N/A
LOG_LEVEL_WARNING	3	Warning	macro definition	N/A
LOG_LEVEL_ERROR	4	Error log	macro definition	N/A
LOG_LEVEL_FATAL	5	Fatal log	macro definition	N/A

3.9 Message Constants

Table 3-13: Message Constants defines constants specific to Message implementation.

Table 3-13: Message Constants

Mnemonic	Value	Description	Type	Units
HMSG_MSG_MAX	65536	Maximum message	macro definition	bytes
CHUNK_SIZE	65536 * 100	Chunk size	macro definition	bytes
HDTN_FLAG_CUSTODY_REQ	1	Custody requested flag	macro definition	N/A
HDTN_FLAG_CUSTODY_OK	2	Custody OK flag	macro definition	N/A
HDTN_FLAG_CUSTODY_FAIL	4	Custody fail flag	macro definition	N/A
HDTN_MSGTYPE_EGRESS	4	Egress message	macro definition	N/A
HDTN_MSGTYPE_STORE	5	Store message	macro definition	N/A
HDTN_MSGTYPE_EGRESS_ADD_OPPORTUNISTIC_LINK	6	Egress and opportunistic link message	macro definition	N/A
HDTN_MSGTYPE_EGRESS_REMOVE_OPPORTUNISTIC_LINK	7	Egress remove opportunistic link message	macro definition	N/A
HDTN_MSGTYPE_STORAGE_ADD_OPPORTUNISTIC_LINK	8	Storage add opportunistic link message	macro definition	N/A
HDTN_MSGTYPE_STORAGE_REMOVE_OPPORTUNISTIC_LINK	9	Storage remove opportunistic link message	macro definition	N/A
HDTN_MSGTYPE_BUNDLES_TO_ROUTER	10	Bundles to router message	macro definition	N/A
HDTN_MSGTYPE_BUNDLES_FROM_ROUTER	11	Bundles from router message	macro definition	N/A
HDTN_MSGTYPE_ENOTIMPL	57,344	Convergence layer type not implemented	macro definition	N/A
HDTN_MSGTYPE_COK	61,440	Acknowledgement that previous command was processed successfully	macro definition	N/A

Mnemonic	Value	Description	Type	Units
HDTN_MSGTYPE_CFAIL	61,441	Negative acknowledgement of previous command	macro definition	N/A
HDTN_MSGTYPE_CTELEM_REQ	61,442	Request for telemetry from the application	macro definition	N/A
HDTN_MSGTYPE_CSCHED_REQ	61,443	Request for a scheduled event	macro definition	N/A
HDTN_MSGTYPE_TSTORAGE	64,256	Response that indicates telemetry is of type “storage”	macro definition	N/A
HDTN_MSGTYPE_IOK	64,512	Indicates successful worker startup	macro definition	N/A
HDTN_MSGTYPE_IABORT	64,513	Indicates that the worker encountered a critical failure and will immediately terminate	macro definition	N/A
HDTN_MSGTYPE_ISHUTDOWN	64,514	Tells the worker to shut down	macro definition	N/A
HDTN_MSGTYPE_ILINKUP	64,515	Link available event from router	macro definition	N/A
HDTN_MSGTYPE_ILINKDOWN	64,516	Link unavailable event from router	macro definition	N/A
HDTN_MSGTYPE_IPRELOAD	64,517	Preloads data because an event is scheduled to begin soon	macro definition	N/A
HDTN_MSGTYPE_IWORKSTATS	64,518	Update on worker stats sent from worker to parent	macro definition	N/A
HDTN_MSGTYPE_ROUTEUPDATE	64,519	Route update event from router process	macro definition	N/A
HDTN_MSGTYPE_LINKSTATUS	64,520	Link status update event from egress process	macro definition	N/A
CPM_NEW_CONTACT_PLAN	64,521	Reload with new contact plan message	macro definition	N/A
HDTN_MSGTYPE_EGRESS_FAILED_BUNDLE_TO_STORAGE	21,844	Egress failed bundle to storage message	macro definition	N/A
HDTN_MSGTYPE_EGRESS_ACK_TO_STORAGE	21,845	Egress acknowledge to storage message	macro definition	N/A
HDTN_MSGTYPE_EGRESS_ACK_TO_INGRESS	21,846	Egress acknowledge to ingress message	macro definition	N/A
HDTN_MSGTYPE_STORAGE_ACK_TO_INGRESS	21,847	Storage acknowledge to ingress message	macro definition	N/A
HDTN_MSGTYPE_ALL_OUTDUCT_CAPABILITIES_TELEMETRY	21,848	All outduct capabilities telemetry message	macro definition	N/A
HDTN_MSGTYPE_DEPLETED_STORAGE_REPORT	21,849	Depleted storage report message	macro definition	N/A
HDTN_NOROUTE	UINT64_MAX	No route available	macro definition	N/A

3.10 Serial Line Internet Protocol Constants

Table 3-14: SLIP Constants defines the constants specific to Serial Line Internet Protocol (SLIP) implementation.

Table 3-14: SLIP Constants

Mnemonic	Value	Description	Type	Units
SLIP_END	0xC0	Frame end	macro definition	N/A
SLIP_ESC	0xDB	Frame escape	macro definition	N/A
SLIP_ESC_END	0xDC	Transposed frame end	macro definition	N/A
SLIP_ESC_ESC	0xDD	Transposed frame escape	macro definition	N/A

3.11 Telemetry Definitions Constants

Table 3-15: Telemetry Definitions Constants defines the constants specific to Telemetry Definitions implementation.

Table 3-15: Telemetry Definitions Constants

Mnemonic	Value	Description	Type	Units
ZMQ_CONNECTION_ID_LEN	5	ZeroMQ connection ID length	uint8	bytes

3.12 UINT8 Padded Vector Constants

Table 3-16: UINT8 Padded Vector Constants defines the constants specific to UINT8 Padded Vector implementation.

Table 3-16: UINT8 Padded Vector Constants

Mnemonic	Value	Description	Type	Units
PADDING_ELEMENTS_BEFORE	256	Padding elements before	size	bytes
PADDING_ELEMENTS_AFTER	32	Padding elements after	size	bytes
TOTAL_PADDING_ELEMENTS	288	Total padding elements	size	bytes

3.13 ZeroMQ Constants

Table 3-17: ZeroMQ Constants define the constants specific to ZeroMQ wrapper implementation.

Table 3-17: ZeroMQ Constants

Mnemonic	Value	Description	Type	Units
CPPZMQ_VERSION_MAJOR	4	CPP ZeroMQ version – major	macro definition	N/A
CPPZMQ_VERSION_MINOR	7	CPP ZeroMQ version – minor	macro definition	N/A
CPPZMQ_VERSION_PATCH	1	CPP ZeroMQ version – patch	macro definition	N/A
ZMQ_MAKE_VERSION	(CPPZMQ_VERSION_MAJOR, CPPZMQ_VERSION_MINOR, CPPZMQ_VERSION_PATCH)	ZeroMQ make version	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the _WIN32 symbolic constant.

Table 3-18: ZeroMQ Constants (_WIN32 Defined) defines the constants specific to ZeroMQ wrapper implementation when _WIN32 is defined.

Table 3-18: ZeroMQ Constants (_WIN32 Defined)

Mnemonic	Value	Description	Type	Units
NOMINMAX	N/A	No min/max compiler directive	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the _MSVC_LANG symbolic constant.

Table 3-19: ZeroMQ Constants (_MSVC_LANG Defined) defines the constants specific to ZeroMQ wrapper implementation when _MSVC_LANG is defined.

Table 3-19: ZeroMQ Constants (_MSVC_LANG Defined)

Mnemonic	Value	Description	Type	Units
CPPZMQ_LANG	_MSVC_LANG	CPP ZeroMQ language	macro definition	N/A

Table 3-20: ZeroMQ Constants (_MSVC_LANG Not Defined) defines the constants specific to ZeroMQ wrapper implementation when _MSVC_LANG is not defined.

Table 3-20: ZeroMQ Constants (_MSVC_LANG Not Defined)

Mnemonic	Value	Description	Type	Units
CPPZMQ_LANG	__cplusplus	CPP ZeroMQ language	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the _HAS_CXX14 and CPPZMQ_LANG symbolic constants.

Table 3-21: ZeroMQ Constants ((_HAS_CXX14 Defined) AND (_HAS_CXX14 != 0) AND (CPPZMQ_LANG < 201402L)) defines the constants specific to ZeroMQ wrapper implementation when _HAS_CXX14 is defined and the following are TRUE:

- _HAS_CXX14 is non-zero
- CPPZMQ_LANG < 201402L

Table 3-21: ZeroMQ Constants ((_HAS_CXX14 Defined) AND (_HAS_CXX14 != 0) AND (CPPZMQ_LANG < 201402L))

Mnemonic	Value	Description	Type	Units
CPPZMQ_LANG	201402L	CPP ZeroMQ language	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the _HAS_CXX17 and CPPZMQ_LANG symbolic constants.

Table 3-22: ZeroMQ Constants ((_HAS_CXX14 Defined) AND (_HAS_CXX17 != 0) AND (CPPZMQ_LANG < 201703L)) defines the constants specific to ZeroMQ wrapper implementation when _HAS_CXX17 is defined and the following are TRUE:

- _HAS_CXX17 is non-zero
- CPPZMQ_LANG < 201703L

Table 3-22: ZeroMQ Constants ((_HAS_CXX14 Defined) AND (_HAS_CXX17 != 0) AND (CPPZMQ_LANG < 201703L))

Mnemonic	Value	Description	Type	Units
CPPZMQ_LANG	201703L	CPP ZeroMQ language	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the CPPZMQ_LANG and _MSC_VER symbolic constants.

Table 3-23: ZeroMQ Constants ((CPPZMQ_LANG ≥ 201103L) OR ((_MSC_VER Defined) AND (_MSC_VER ≥ 1900))) defines the constants specific to ZeroMQ wrapper implementation when one or more of the following are TRUE:

- CPPZMQ_LANG ≥ 201103L
- _MSC_VER is defined AND _MSC_VER ≥ 1900

Table 3-23: ZeroMQ Constants ((CPPZMQ_LANG ≥ 201103L) OR ((_MSC_VER Defined) AND (_MSC_VER ≥ 1900)))

Mnemonic	Value	Description	Type	Units
ZMQ_CPP11	N/A	ZeroMQ CPP11	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the CPPZMQ_LANG symbolic constant.

Table 3-24: ZeroMQ Constants (CPPZMQ_LANG ≥ 201402L) defines the constants specific to ZeroMQ wrapper implementation when CPPZMQ_LANG ≥ 201402L.

Table 3-24: ZeroMQ Constants (CPPZMQ_LANG ≥ 201402L)

Mnemonic	Value	Description	Type	Units
ZMQ_CPP14	N/A	ZeroMQ CPP14	macro definition	N/A

Table 3-25: ZeroMQ Constants (CPPZMQ_LANG ≥ 201703L) defines the constants specific to ZeroMQ wrapper implementation when CPPZMQ_LANG ≥ 201703L.

Table 3-25: ZeroMQ Constants (CPPZMQ_LANG ≥ 201703L)

Mnemonic	Value	Description	Type	Units
ZMQ_CPP17	N/A	ZeroMQ CPP17	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the ZMQ_CPP14, _MSC_VER, and __GNUC__ symbolic constants.

Table 3-26: ZeroMQ Constants (ZMQ_CPP14 Defined AND _MSC_VER Not Defined) defines the constants specific to ZeroMQ wrapper implementation when ZMQ_CPP14 is defined and _MSC_VER is not defined.

Table 3-26: ZeroMQ Constants (ZMQ_CPP14 Defined AND _MSC_VER Not Defined)

Mnemonic	Value	Description	Type	Units
ZMQ_DEPRECATED(msg)	[[deprecated(msg)]]	ZeroMQ deprecated	macro definition	N/A

Table 3-27: ZeroMQ Constants (ZMQ_CPP14 Not Defined AND _MSC_VER Defined) defines the constants specific to ZeroMQ wrapper implementation when ZMQ_CPP14 is not defined and _MSC_VER is defined.

Table 3-27: ZeroMQ Constants (ZMQ_CPP14 Not Defined AND _MSC_VER Defined)

Mnemonic	Value	Description	Type	Units
ZMQ_DEPRECATED(msg)	__declspec(deprecated(msg))	ZeroMQ deprecated	macro definition	N/A

Table 3-28: ZeroMQ Constants (ZMQ_CPP14 Not Defined AND _MSC_VER Not Defined AND __GNUC__ Defined) defines the constants specific to ZeroMQ wrapper implementation when ZMQ_CPP14 is not defined, _MSC_VER is not defined, and __GNUC__ is defined.

Table 3-28: ZeroMQ Constants (ZMQ_CPP14 Not Defined AND _MSC_VER Not Defined AND __GNUC__ Defined)

Mnemonic	Value	Description	Type	Units
ZMQ_DEPRECATED(msg)	__attribute__((deprecated(msg)))	ZeroMQ deprecated	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the ZMQ_CPP17 symbolic constant.

Table 3-29: ZeroMQ Constants (ZMQ_CPP17 Defined) defines the constants specific to ZeroMQ wrapper implementation when ZMQ_CPP17 is defined.

Table 3-29: ZeroMQ Constants (ZMQ_CPP17 Defined)

Mnemonic	Value	Description	Type	Units
ZMQ_NODISCARD	[[nodiscard]]	ZeroMQ no discard	macro definition	N/A
ZMQ_INLINE_VAR	inline	ZeroMQ inline variable	macro definition	N/A

Table 3-30: ZeroMQ Constants (ZMQ_CPP17 Not Defined) defines the constants specific to ZeroMQ wrapper implementation when ZMQ_CPP17 is not defined.

Table 3-30: ZeroMQ Constants (ZMQ_CPP17 Not Defined)

Mnemonic	Value	Description	Type	Units
ZMQ_NODISCARD	N/A	ZeroMQ no discard	macro definition	N/A
ZMQ_INLINE_VAR	N/A	ZeroMQ inline variable	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the ZMQ_CPP11 symbolic constant.

Table 3-31: ZeroMQ Constants (ZMQ_CPP11 Defined) defines the constants specific to ZeroMQ wrapper implementation when ZMQ_CPP11 is defined.

Table 3-31: ZeroMQ Constants (ZMQ_CPP11 Defined)

Mnemonic	Value	Description	Type	Units
ZMQ_NOTHROW	noexcept	ZeroMQ no throw	macro definition	N/A
ZMQ_EXPLICIT	explicit	ZeroMQ explicit	macro definition	N/A
ZMQ_OVERRIDE	override	ZeroMQ override	macro definition	N/A
ZMQ_NULLPTR	nullptr	ZeroMQ null pointer	macro definition	N/A
ZMQ_CONSTEXPR_FN	constexpr	ZeroMQ constant expression – function	macro definition	N/A
ZMQ_CONSTEXPR_VAR	constexpr	ZeroMQ constant expression – variable	macro definition	N/A
ZMQ_CPP11_DEPRECATED(msg)	ZMQ_DEPRECATED(msg)	ZeroMQ CPP11 deprecated	macro definition	N/A

Table 3-32: ZeroMQ Constants (ZMQ_CPP11 Not Defined) defines the constants specific to ZeroMQ wrapper implementation when ZMQ_CPP11 is not defined.

Table 3-32: ZeroMQ Constants (ZMQ_CPP11 Not Defined)

Mnemonic	Value	Description	Type	Units
ZMQ_NOTHROW	throw()	ZeroMQ no throw	macro definition	N/A
ZMQ_EXPLICIT	N/A	ZeroMQ explicit	macro definition	N/A
ZMQ_OVERRIDE	N/A	ZeroMQ override	macro definition	N/A
ZMQ_NULLPTR	0	ZeroMQ null pointer	macro definition	N/A
ZMQ_CONSTEXPR_FN	N/A	ZeroMQ constant expression – function	macro definition	N/A
ZMQ_CONSTEXPR_VAR	const	ZeroMQ constant expression – variable	macro definition	N/A
ZMQ_CPP11_DEPRECATED(msg)	N/A	ZeroMQ CPP11 deprecated	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the ZMQ_CPP14 and _MSC_VER symbolic constants.

Table 3-33: ZeroMQ Constants (ZMQ_CPP14 Defined AND (_MSC_VER Not Defined OR _MSC_VER > 1900)) defines the constants specific to ZeroMQ wrapper implementation when the following are TRUE:

- ZMQ_CPP14 is defined
- _MSC_VER is not defined OR _MSC_VER > 1900

Table 3-33: ZeroMQ Constants (ZMQ_CPP14 Defined AND (_MSC_VER Not Defined OR _MSC_VER > 1900))

Mnemonic	Value	Description	Type	Units
ZMQ_EXTENDED_CONSTEXPR	N/A	ZeroMQ extended constant expression	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the __has_include and ZMQ_CPP17 symbolic constants.

Table 3-34: ZeroMQ Constants (__has_include Defined AND ZMQ_CPP17 Defined) defines the constants specific to ZeroMQ wrapper implementation when __has_include and ZMQ_CPP17 are both defined.

Table 3-34: ZeroMQ Constants (__has_include Defined AND ZMQ_CPP17 Defined)

Mnemonic	Value	Description	Type	Units
CPPZMQ_HAS_INCLUDE_CPP17(X)	__has_include(X)	CPP ZeroMQ has include CPP17	macro definition	N/A

Table 3-35: ZeroMQ Constants (__has_include Not Defined OR ZMQ_CPP17 Not Defined) defines the constants specific to ZeroMQ wrapper implementation when __has_include is not defined OR ZMQ_CPP17 is not defined.

Table 3-35: ZeroMQ Constants (__has_include Not Defined OR ZMQ_CPP17 Not Defined)

Mnemonic	Value	Description	Type	Units
CPPZMQ_HAS_INCLUDE_CPP17(X)	0	CPP ZeroMQ has include CPP17	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the CPPZMQ_HAS_INCLUDE_CPP17 and CPPZMQ_HAS_OPTIONAL symbolic constants.

Table 3-36: ZeroMQ Constants (CPPZMQ_HAS_INCLUDE_CPP17(<optional>) Defined AND CPPZMQ_HAS_OPTIONAL Not Defined) defines the constants specific to ZeroMQ wrapper implementation when CPPZMQ_HAS_INCLUDE_CPP17(<optional>) is defined and CPPZMQ_HAS_OPTIONAL is not defined.

Table 3-36: ZeroMQ Constants (CPPZMQ_HAS_INCLUDE_CPP17(<optional>) Defined AND CPPZMQ_HAS_OPTIONAL Not Defined)

Mnemonic	Value	Description	Type	Units
CPPZMQ_HAS_OPTIONAL	1	CPP ZeroMQ has optional	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the CPPZMQ_HAS_OPTIONAL symbolic constant.

Table 3-37: ZeroMQ Constants (CPPZMQ_HAS_OPTIONAL Not Defined) defines the constants specific to ZeroMQ wrapper implementation when CPPZMQ_HAS_OPTIONAL is not defined.

Table 3-37: ZeroMQ Constants (CPPZMQ_HAS_OPTIONAL Not Defined)

Mnemonic	Value	Description	Type	Units
CPPZMQ_HAS_OPTIONAL	0	CPP ZeroMQ has optional	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the CPPZMQ_HAS_INCLUDE_CPP17 and CPPZMQ_HAS_STRING_VIEW symbolic constants.

Table 3-38: ZeroMQ Constants (CPPZMQ_HAS_INCLUDE_CPP17(<string view>) Defined AND CPPZMQ_HAS_STRING_VIEW Not Defined) defines the constants specific to ZeroMQ wrapper implementation when CPPZMQ_HAS_INCLUDE_CPP17(<string view>) is defined and CPPZMQ_HAS_STRING_VIEW is not defined.

Table 3-38: ZeroMQ Constants (CPPZMQ_HAS_INCLUDE_CPP17(<string view>) Defined AND CPPZMQ_HAS_STRING_VIEW Not Defined)

Mnemonic	Value	Description	Type	Units
CPPZMQ_HAS_STRING_VIEW	1	CPP ZeroMQ has string view	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the CPPZMQ_HAS_STRING_VIEW symbolic constant.

Table 3-39: ZeroMQ Constants (CPPZMQ_HAS_STRING_VIEW Not Defined) defines the constants specific to ZeroMQ wrapper implementation when CPPZMQ_HAS_STRING_VIEW is not defined.

Table 3-39: ZeroMQ Constants (CPPZMQ_HAS_STRING_VIEW Not Defined)

Mnemonic	Value	Description	Type	Units
CPPZMQ_HAS_STRING_VIEW	0	CPP ZeroMQ has string view	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the __GNUC__, __GNUC_MINOR__, __GXX_EXPERIMENTAL_CXX0X__, __clang__, __has_feature, and _MSC_VER symbolic constants.

Table 3-40: ZeroMQ Constants (__GNUC__ Defined AND __GXX_EXPERIMENTAL_CXX0X__ Defined AND (__GNUC__ > 4 OR (__GNUC__ = 4 AND __GNUC_MINOR__ > 2))) defines the constants specific to ZeroMQ wrapper implementation when __GNUC__ and __GXX_EXPERIMENTAL_CXX0X__ are both defined and either of the following is TRUE:

- __GNUC__ > 4
- __GNUC__ = 4 AND __GNUC_MINOR__ > 2

Table 3-40: ZeroMQ Constants (__GNUC__ Defined AND __GXX_EXPERIMENTAL_CXX0X__ DefinedAND (__GNUC__ > 4 OR (__GNUC__ = 4 AND __GNUC_MINOR__ > 2)))

Mnemonic	Value	Description	Type	Units
ZMQ_HAS_RVALUE_REFS	N/A	ZeroMQ has RValue references	macro definition	N/A
ZMQ_DELETED_FUNCTION	delete	ZeroMQ deleted function	macro definition	N/A

Table 3-41: ZeroMQ Constants ((__GNUC__ NOT Defined OR __GNUC__ < 4 OR (__GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2) OR __GXX_EXPERIMENTAL_CXX0X__ NOT Defined) AND __clang__ Defined AND __has_feature(cxx_rvalue_references) Defined) defines the constants specific to ZeroMQ wrapper implementation when the following are TRUE:

- any of the following are TRUE:
    - __GNUC__ is not defined
    - __GNUC__ < 4
    - __GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2
    - __GXX_EXPERIMENTAL_CXX0X__ is not defined
- __clang__ is defined
- __has_feature(cxx_rvalue_references) is defined

Table 3-41: ZeroMQ Constants ((__GNUC__ NOT Defined OR __GNUC__ < 4 OR (__GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2) OR __GXX_EXPERIMENTAL_CXX0X__ NOT Defined) AND __clang__ Defined AND __has_feature(cxx_rvalue_references) Defined)

Mnemonic	Value	Description	Type	Units
ZMQ_HAS_RVALUE_REFS	N/A	ZeroMQ has RValue references	macro definition	N/A

Table 3-42: ZeroMQ Constants ((__GNUC__ NOT Defined OR __GNUC__ < 4 OR (__GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2) OR __GXX_EXPERIMENTAL_CXX0X__ NOT Defined) AND __clang__ Defined AND __has_feature(cxx_deleted_functions) Defined) defines the constants specific to ZeroMQ wrapper implementation when the following are TRUE:

- any of the following are TRUE:
    - __GNUC__ is not defined
    - __GNUC__ < 4
    - __GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2
    - __GXX_EXPERIMENTAL_CXX0X__ is not defined
- __clang__ is defined
- __has_feature(cxx_deleted_functions) is defined

Table 3-42: ZeroMQ Constants ((__GNUC__ NOT Defined OR __GNUC__ < 4 OR (__GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2) OR __GXX_EXPERIMENTAL_CXX0X__ NOT Defined) AND __clang__ Defined AND __has_feature(cxx_deleted_functions) Defined)

Mnemonic	Value	Description	Type	Units
ZMQ_DELETED_FUNCTION	delete	ZeroMQ has RValue references	macro definition	N/A

Table 3-43: ZeroMQ Constants ((__GNUC__ NOT Defined OR __GNUC__ < 4 OR (__GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2) OR __GXX_EXPERIMENTAL_CXX0X__ NOT Defined) AND __clang__ Not Defined AND _MSC_VER Defined AND _MSC_VER ≥ 1900) defines the constants specific to ZeroMQ wrapper implementation when the following are TRUE:

- any of the following are TRUE:
    - __GNUC__ is not defined
    - __GNUC__ < 4
    - __GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2
    - __GXX_EXPERIMENTAL_CXX0X__ is not defined
- __clang__ is not defined
- _MSC_VER is defined
- _MSC_VER ≥ 1900

Table 3-43: ZeroMQ Constants ((__GNUC__ NOT Defined OR __GNUC__ < 4 OR (__GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2) OR __GXX_EXPERIMENTAL_CXX0X__ NOT Defined) AND __clang__ Not Defined AND _MSC_VER Defined AND _MSC_VER ≥ 1900)

Mnemonic	Value	Description	Type	Units
ZMQ_HAS_RVALUE_REFS	N/A	ZeroMQ has RValue references	macro definition	N/A
ZMQ_DELETED_FUNCTION	delete	ZeroMQ deleted function	macro definition	N/A

Table 3-44: ZeroMQ Constants ((__GNUC__ NOT Defined OR __GNUC__ < 4 OR (__GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2) OR __GXX_EXPERIMENTAL_CXX0X__ NOT Defined) AND __clang__ Not Defined AND _MSC_VER Defined AND _MSC_VER < 1900 AND _MSC_VER ≥ 1600) defines the constants specific to ZeroMQ wrapper implementation when the following are TRUE:

- any of the following are TRUE:
    - __GNUC__ is not defined
    - __GNUC__ < 4
    - __GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2
    - __GXX_EXPERIMENTAL_CXX0X__ is not defined
- __clang__ is not defined
- _MSC_VER is defined
- _MSC_VER < 1900
- _MSC_VER ≥ 1600

Table 3-44: ZeroMQ Constants ((__GNUC__ NOT Defined OR __GNUC__ < 4 OR (__GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2) OR __GXX_EXPERIMENTAL_CXX0X__ NOT Defined) AND __clang__ Not Defined AND _MSC_VER Defined AND _MSC_VER < 1900 AND _MSC_VER ≥ 1600)

Mnemonic	Value	Description	Type	Units
ZMQ_HAS_RVALUE_REFS	N/A	ZeroMQ has RValue references	macro definition	N/A
ZMQ_DELETED_FUNCTION	N/A	ZeroMQ deleted function	macro definition	N/A

Table 3-45: ZeroMQ Constants ((__GNUC__ NOT Defined OR __GNUC__ < 4 OR (__GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2) OR __GXX_EXPERIMENTAL_CXX0X__ NOT Defined) AND __clang__ Not Defined AND (_MSC_VER NOT Defined OR _MSC_VER < 1600)) defines the constants specific to ZeroMQ wrapper implementation when the following are TRUE:

- any of the following are TRUE:
  - __GNUC__ is not defined
  - __GNUC__ < 4
  - __GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2
  - __GXX_EXPERIMENTAL_CXX0X__ is not defined
- __clang__ is not defined
- any of the following are TRUE:
  - _MSC_VER is not defined
  - _MSC_VER < 1600

Table 3-45: ZeroMQ Constants ((__GNUC__ NOT Defined OR __GNUC__ < 4 OR (__GNUC__ = 4 AND __GNUC_MINOR__ ≤ 2) OR __GXX_EXPERIMENTAL_CXX0X__ NOT Defined) AND __clang__ Not Defined AND (_MSC_VER NOT Defined OR _MSC_VER < 1600))

Mnemonic	Value	Description	Type	Units
ZMQ_DELETED_FUNCTION	N/A	ZeroMQ deleted function	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the ZMQ_CPP11, __llvm__, __INTEL_COMPILER, __GNUC__, and __GLIBCXX__ symbolic constants.

Table 3-46: ZeroMQ Constants (ZMQ_CPP11 Defined AND __llvm__ Not Defined AND __INTEL_COMPILER Not Defined AND __GNUC__ Defined AND __GNUC__ < 5) defines the constants specific to ZeroMQ wrapper implementation when the following are TRUE:

- ZMQ_CPP11 is defined
- __llvm__ is not defined
- __INTEL_COMPILER is not defined
- __GNUC__ is defined
- __GNUC__ < 5

Table 3-46: ZeroMQ Constants (ZMQ_CPP11 Defined AND __llvm__ Not Defined AND __INTEL_COMPILER Not Defined AND __GNUC__ Defined AND __GNUC__ < 5)

Mnemonic	Value	Description	Type	Units
ZMQ_CPP11_PARTIAL	N/A	ZeroMQ CPP11 partial	macro definition	N/A

Table 3-47: ZeroMQ Constants ((ZMQ_CPP11 Not Defined OR __llvm__ Defined OR __INTEL_COMPILER Defined OR __GNUC__ Not Defined OR __GNUC__ ≥ 5) AND __GLIBCXX__ Defined AND __GLIBCXX__ < 20160805) defines the constants specific to ZeroMQ wrapper implementation when the following are TRUE:

- any of the following are TRUE:
  - ZMQ_CPP11 is not defined
  - __llvm__ is defined
  - __INTEL_COMPILER is defined
  - __GNUC__ is not defined
  - __GNUC__ ≥ 5

- __GLIBCXX__ is defined
- __GLIBCXX__ < 20160805

Table 3-47: ZeroMQ Constants ((ZMQ_CPP11 Not Defined OR __llvm__ Defined OR __INTEL_COMPILER Defined OR __GNUC__ Not Defined OR __GNUC__ ≥ 5) AND __GLIBCXX__ Defined AND __GLIBCXX__ < 20160805)

Mnemonic	Value	Description	Type	Units
ZMQ_CPP11_PARTIAL	N/A	ZeroMQ CPP11 partial	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the ZMQ_CPP11 and ZMQ_CPP11_PARTIAL symbolic constants.

Table 3-48: ZeroMQ Constants (ZMQ_CPP11 Defined AND ZMQ_CPP11_PARTIAL Defined) defines the constants specific to ZeroMQ wrapper implementation when ZMQ_CPP11 and ZMQ_CPP11_PARTIAL are defined.

Table 3-48: ZeroMQ Constants (ZMQ_CPP11 Defined AND ZMQ_CPP11_PARTIAL Defined)

Mnemonic	Value	Description	Type	Units
ZMQ_IS_TRIVIALLY_COPYABLE(T)	__has_trivial_copy(T)	ZeroMQ is trivialy copyable	macro definition	N/A

Table 3-49: ZeroMQ Constants (ZMQ_CPP11 Defined AND ZMQ_CPP11_PARTIAL Not Defined) defines the constants specific to ZeroMQ wrapper implementation when ZMQ_CPP11 is defined and ZMQ_CPP11_PARTIAL is not defined.

Table 3-49: ZeroMQ Constants (ZMQ_CPP11 Defined AND ZMQ_CPP11_PARTIAL Not Defined)

Mnemonic	Value	Description	Type	Units
ZMQ_IS_TRIVIALLY_COPYABLE(T)	std::is_trivially_copyable<T>::value	ZeroMQ is trivialy copyable	macro definition	N/A

Definition of the following ZeroMQ constants is dependent on definition of the ZMQ_VERSION symbolic constant.

Table 3-50: ZeroMQ Constants (ZMQ_VERSION ≥ ZMQ_MAKE_VERSION(3, 3, 0)) defines the constants specific to ZeroMQ wrapper implementation when ZMQ_VERSION ≥ ZMQ_MAKE_VERSION(3, 3, 0).

Table 3-50: ZeroMQ Constants (ZMQ_VERSION ≥ ZMQ_MAKE_VERSION(3, 3, 0))

Mnemonic	Value	Description	Type	Units
ZMQ_NEW_MONITOR_EVENT_LAYOUT	N/A	ZeroMQ new monitor event layout	macro definition	N/A

Table 3-51: ZeroMQ Constants (ZMQ_VERSION ≥ ZMQ_MAKE_VERSION(4, 1, 0)) defines the constants specific to ZeroMQ wrapper implementation when ZMQ_VERSION ≥ ZMQ_MAKE_VERSION(4, 1, 0).

Table 3-51: ZeroMQ Constants (ZMQ_VERSION ≥ ZMQ_MAKE_VERSION(4, 1, 0))

Mnemonic	Value	Description	Type	Units
ZMQ_HAS_PROXY_STEERABLE	N/A	ZeroMQ has proxy steerable	macro definition	N/A

4.0 Enumeration Data

The following section defines enumerated integral values used in HDTN software.

4.1 Boost Enumerations

The following section defines enumerations specific to Boost implementation.

4.1.1 Boost Event Type

Table 4-1: Boost Event Type defines Boost Event Type enumerated 32-bit integer values.

Table 4-1: Boost Event Type

Mnemonic	Value	Description
unassigned	<0	Unassigned
null	0	Null
added	1	Directory added
removed	2	Directory removed
modified	3	Directory modified
renamed_old_name	4	Directory renamed (old name)
renamed_new_name	5	Directory renamed (new name)
recursive_rescan	6	Rescan directory
unassigned	>6	Unassigned

4.2 Bundle Protocol Receive Stream Enumerations

This section defines enumerations specific to Bundle Protocol Receive (BPReceive) Stream implementation.

4.2.1 BPReceive Stream Outduct Types

Table 4-2: BPReceive Stream Outduct Types defines BPReceive Stream Outduct Types enumerated 32-bit integer values.

Table 4-2: BPReceive Stream Outduct Types

Mnemonic	Value	Description
unassigned	<0	Unassigned
UDP_OUTDUCT	0	UDP outduct
GSTREAMER_APPSRC_OUTDUCT	1	GStreamer application source outduct
unassigned	>1	Unassigned

4.3 Bundle Protocol Security Enumerations

This section defines enumerations specific to Bundle Protocol Security (BPSec) implementation.

4.3.1 BPSec BIB-HMAC-SHA2 Integrity Scope Flags

Table 4-3: BPSec BIB-HMAC-SHA2 Integrity Scope Flags defines BPSec Block Integrity Block Hash-based Message Authentication Code Secure Hash Algorithm 2 (BIB-HMAC-SHA2) Integrity Scope Flags enumerated 32-bit integer values.

Table 4-3: BPSec BIB-HMAC-SHA2 Integrity Scope Flags

Mnemonic	Value	Description
unassigned	<0	Unassigned
INCLUDE_PRIMARY_BLOCK_FLAG	0	Include primary block flag
INCLUDE_TARGET_HEADER_FLAG	1	Include target header flag
INCLUDE_SECURITY_HEADER_FLAG	2	Include security header flag
unassigned	3-7	Reserved
unassigned	>7	Unassigned

4.3.2 BPSec BIB-HMAC-SHA2 Security Parameters

Table 4-4: BPSec BIB-HMAC-SHA2 Security Parameters defines BPSec BIB-HMAC-SHA2 Security Parameters enumerated 32-bit integer values.

Table 4-4: BPSec BIB-HMAC-SHA2 Security Parameters

Mnemonic	Value	Description
unassigned	<1	Unassigned
SHA_VARIANT	1	SHA variant
WRAPPED_KEY	2	Wrapped key
INTEGRITY_SCOPE_FLAGS	3	Integrity scope flags
unassigned	>3	Unassigned

4.3.3 BPSec BIB-HMAC-SHA2 Security Results

Table 4-5: BPSec BIB-HMAC-SHA2 Security Results defines BPSec BIB-HMAC-SHA2 Security Results enumerated 32-bit integer values.

Table 4-5: BPSec BIB-HMAC-SHA2 Security Results

Mnemonic	Value	Description
unassigned	<1	Unassigned
EXPECTED_HMAC	1	Expected hash message authentication code
unassigned	>1	Unassigned

4.3.4 BPSec BCB-AES-GCM Additional Authentication Data Scope Flags

Table 4-6: BPSec BCB-AES-GCM Additional Authentication Data Scope Flags defines BPSec Block Confidentiality Block Advanced Encryption Standard Galois/Counter Mode (BCB-AES-GCM) Additional Authentication Data Scope Flags enumerated 32-bit integer values.

Table 4-6: BPSec BCB-AES-GCM Additional Authentication Data Scope Flags

Mnemonic	Value	Description
unassigned	<0	Unassigned
INCLUDE_PRIMARY_BLOCK_FLAG	0	Include primary block flag
INCLUDE_TARGET_HEADER_FLAG	1	Include target header flag
INCLUDE_SECURITY_HEADER_FLAG	2	Include security header flag
unassigned	3-7	Reserved
unassigned	>7	Unassigned

4.3.5 BPSec BCB-AES-GCM Additional Authentication Data Security Parameters

Table 4-7: BPSec BIB-HMAC-SHA2 Security Parameters defines BPSec BIB-HMAC-SHA2 Security Parameters enumerated 32-bit integer values.

Table 4-7: BPSec BIB-HMAC-SHA2 Security Parameters

Mnemonic	Value	Description
unassigned	<1	Unassigned
INITIALIZATION_VECTOR	1	Initialization vector
AES_VARIANT	2	AES variant
WRAPPED_KEY	3	Wrapped key
AAD_SCOPE_FLAGS	4	Additional Authenticated Data (AAD) scope flags
unassigned	>4	Unassigned

4.3.6 BPSec BCB-AES-GCM Additional Authentication Data Security Results

Table 4-8: BPSec BCB-AES-GCM Additional Authentication Data Security Results defines BPSec BCB-AES-GCM Additional Authentication Data Security Results enumerated 32-bit integer values.

Table 4-8: BPSec BCB-AES-GCM Additional Authentication Data Security Results

Mnemonic	Value	Description
unassigned	<1	Unassigned
AUTHENTICATION_TAG	1	Authentication tag
unassigned	>1	Unassigned

4.3.7 BPSec Security Context Identifiers

Table 4-9: BPSec Security Context Identifiers defines BPSec Security Context Identifiers enumerated 32-bit integer values.

Table 4-9: BPSec Security Context Identifiers

Mnemonic	Value	Description
unassigned	<1	Unassigned
BIB_HMAC_SHA2	1	Bundle Integrity Block Hash-based Message Authentication Code Secure Hash Algorithm 2
BCB_AES_GCM	2	Bundle Confidentiality Block Advanced Encryption Standard Galois/Counter Mode
unassigned	>2	Unassigned

4.4 Bundle Protocol Security Policy Manager Enumerations

This section defines enumerations specific to BPSec Policy Manager implementation.

4.4.1 BPSec Role

Table 4-10: BPSec Role defines BPSec Role enumerated 32-bit integer values.

Table 4-10: BPSec Role

Mnemonic	Value	Description
unassigned	<0	Unassigned
SOURCE	0	Source
VERIFIER	1	Verifier
ACCEPTOR	2	Acceptor
RESERVED_MAX_ROLE_TYPES	3	Reserved max role types
unassigned	>3	Unassigned

4.5 Bundle Protocol Send Stream Enumerations

This section defines enumerations specific to Bundle Protocol Send (BPSend) Stream implementation.

4.5.1 BPSend Stream Intake Types

Table 4-11: BPSend Stream Intake Types defines BPSend Stream Intake Types enumerated 32-bit integer values.

Table 4-11: BPSend Stream Intake Types

Mnemonic	Value	Description
unassigned	<0	Unassigned
HDTN_APPSINK_INTAKE	0	HDTN application sink intake
HDTN_UDP_INTAKE	1	HDTN UDP intake
HDTN_SHM_INTAKE	2	HDTN SHM intake
unassigned	>2	Unassigned

4.6 Bundle Protocol Version 6 Enumerations

This section defines enumerations specific to BPv6 implementation.

4.6.1 BPv6 Administrative Record Type Code

Table 4-12: BPv6 Administrative Record Type Code defines BPv6 Administrative Record Type Code enumerated unsigned 8-bit integer values.

Table 4-12: BPv6 Administrative Record Type Code

Mnemonic	Value	Description
UNUSED_ZERO	0	Unused
BUNDLE_STATUS_REPORT	1	Bundle status report
CUSTODY_SIGNAL	2	Custody signal
unassigned	3	Unassigned
AGGREGATE_CUSTODY_SIGNAL	4	Aggregate custody signal
unassigned	5-6	Unassigned
ENCAPSULATED_BUNDLE	7	Encapsulated bundle
unassigned	8-41	Unassigned
SAGA_MESSAGE	42	Saga message
unassigned	>42	Unassigned

4.6.2 BPv6 Aggregate Custody Signal Status Reason Indices

Table 4-13: BPv6 ACS Status Reason Indices defines BPv6 Aggregate Custody Signal (ACS) Status Reason Indices enumerated unsigned 8-bit integer values.

Table 4-13: BPv6 ACS Status Reason Indices

Mnemonic	Value	Description
SUCCESS__NO_ADDITIONAL_INFORMATION	0	Success: no additional information
FAIL__REDUNDANT_RECEPTION	1	Fail: redundant reception
FAIL__DEPLETED_STORAGE	2	Fail: depleted storage
FAIL__DESTINATION_ENDPOINT_ID_UNINTELLIGIBLE	3	Fail: destination endpoint ID unintelligible
FAIL__NO_KNOWN_ROUTE_TO_DESTINATION_FROM_HERE	4	Fail: no known route to destination from here

Mnemonic	Value	Description
FAIL__NO_TIMELY_CONTACT_WITH_NEXT_NODE_ON_ROUTE	5	Fail: no timely contact with next node on route
FAIL__BLOCK_UNINTELLIGIBLE	6	Fail: block unintelligible
NUM_INDICES	7	Number of BPv6 status reason indices
unassigned	>7	Unassigned

4.6.3 BPv6 Block Type

Table 4-14: BPv6 Block Type defines BPv6 Block Type enumerated unsigned 8-bit integer values.

Table 4-14: BPv6 Block Type

Mnemonic	Value	Description
PRIMARY_IMPLICIT_ZERO	0	Reserved
PAYLOAD	1	Bundle payload block
BUNDLE_AUTHENTICATION	2	Bundle authentication block
PAYLOAD_INTEGRITY	3	Payload integrity block
PAYLOAD_CONFIDENTIALITY	4	Payload confidentiality block
PREVIOUS_HOP_INSERTION	5	Previous-hop insertion block
UNUSED_6	6	Unused
UNUSED_7	7	Unused
METADATA_EXTENSION	8	Metadata extension block
EXTENSION_SECURITY	9	Extension security block
CUSTODY_TRANSFER_ENHANCEMENT	10	Hop count
UNUSED_11	11	Unused
UNUSED_12	12	Unused
BPLIB_BIB	13	BPLib Bundle-In-Bundle
unassigned	14-19	Unassigned
BUNDLE_AGE	20	Bundle age?
RESERVED_MAX_BLOCK_TYPES	21	Reserved max block types
unassigned	>21	Unassigned

4.6.4 BPv6 Bundle Status Report Reason Codes

Table 4-15: BPv6 Bundle Status Report Reason Codes defines BPv6 Bundle Status Report Reason Codes enumerated unsigned 8-bit integer values.

Table 4-15: BPv6 Bundle Status Report Reason Codes

Mnemonic	Value	Description
NO_ADDITIONAL_INFORMATION	0	No additional information
LIFETIME_EXPIRED	1	Lifetime expired
FORWARDED_OVER_UNIDIRECTIONAL_LINK	2	Forwarded over unidirectional link
TRANSMISSION_CANCELLED	3	Transmission canceled
DEPLETED_STORAGE	4	Depleted storage
DESTINATION_ENDPOINT_ID_UNINTELLIGIBLE	5	Destination endpoint ID unintelligible
NO_KNOWN_ROUTE_TO_DESTINATION_FROM_HERE	6	No known route to destination from here
NO_TIMELY_CONTACT_WITH_NEXT_NODE_ON_ROUTE	7	No timely contact with next node on route
BLOCK_UNINTELLIGIBLE	8	Block unintelligible
unassigned	>8	Unassigned

4.6.5 BPv6 Custody Signal Reason Codes

Table 4-16: BPv6 Custody Signal Reason Codes defines BPv6 Custody Signal Reason Codes enumerated unsigned 8-bit integer values.

Table 4-16: BPv6 Custody Signal Reason Codes

Mnemonic	Value	Description
NO_ADDITIONAL_INFORMATION	0	No additional information
unassigned	1-2	Unassigned
REDUNDANT_RECEPTION	3	Redundant reception
DEPLETED_STORAGE	4	Depleted storage
DESTINATION_ENDPOINT_ID_UNINTELLIGIBLE	5	Destination endpoint ID unintelligible
NO_KNOWN_ROUTE_TO_DESTINATION_FROM_HERE	6	No known route to destination from here
NO_TIMELY_CONTACT_WITH_NEXT_NODE_ON_ROUTE	7	No timely contact with next node on route
BLOCK_UNINTELLIGIBLE	8	Block unintelligible
unassigned	>8	Unassigned

4.6.6 BPv6 Metadata Type Code

Table 4-17: BPv6 Metadata Type Code defines BPv6 Metadata Type Code enumerated unsigned 64-bit integer values.

Table 4-17: BPv6 Metadata Type Code

Mnemonic	Value	Description
UNDEFINED_ZERO	0	Undefined
URI	1	Uniform Resource Identifier endpoint ID
unassigned	>1	Unassigned

4.6.7 BPv6 Priority

Table 4-18: BPv6 Priority defines BPv6 Priority enumerated unsigned 64-bit integer values.

Table 4-18: BPv6 Priority

Mnemonic	Value	Description
BULK	0	Bulk (lowest) priority
NORMAL	1	Normal priority
EXPEDITED	2	Expedited (highest) priority
unassigned	>2	Unassigned

4.6.8 BPv6 Status Reason Indices

Table 4-19: BPv6 Status Reason Indices defines BPv6 Status Reason Indices enumerated unsigned 8-bit integer values.

Table 4-19: BPv6 Status Reason Indices

Mnemonic	Value	Description
SUCCESS__NO_ADDITIONAL_INFORMATION	0	Success: no additional information
FAIL__REDUNDANT_RECEPTION	1	Fail: redundant reception
FAIL__DEPLETED_STORAGE	2	Fail: depleted storage
FAIL__DESTINATION_ENDPOINT_ID_UNINTELLIGIBLE	3	Fail: destination endpoint ID unintelligible
FAIL__NO_KNOWN_ROUTE_TO_DESTINATION_FROM_HERE	4	Fail: no known route to destination from here

Mnemonic	Value	Description
FAIL__NO_TIMELY_CONTACT_WITH_NEXT_NODE_ON_ROUTE	5	Fail: no timely contact with next node on route
FAIL__BLOCK_UNINTELLIGIBLE	6	Fail: block unintelligible
NUM_INDICES	7	Number of BPv6 status reason indices
unassigned	>7	Unassigned

4.7 Bundle Protocol Version 7 Enumerations

This section defines enumerations specific to Bundle Protocol version 7 (BPv7) implementation.

4.7.1 BPv7 Administrative Record Type Code

Table 4-20: BPv7 Administrative Record Type Code defines BPv7 Administrative Record Type Code enumerated unsigned 64-bit integer values.

Table 4-20: BPv7 Administrative Record Type Code

Mnemonic	Value	Description
UNUSED_ZERO	0	Unused
BUNDLE_STATUS_REPORT	1	Bundle status report
unassigned	2	Unassigned
BIBE_PDU	3	Bundle-in-Bundle Encapsulation Protocol Data Unit
CUSTODY_SIGNAL	4	Custody signal
unassigned	>4	Unassigned

4.7.2 BPv7 Block Type Code

Table 4-21: BPv7 Block Type Code defines BPv7 Block Type Code enumerated unsigned 8-bit integer values.

Table 4-21: BPv7 Block Type Code

Mnemonic	Value	Description
PRIMARY_IMPLICIT_ZERO	0	Reserved
PAYLOAD	1	Bundle payload block
UNUSED_2	2	Unused
UNUSED_3	3	Unused
UNUSED_4	4	Unused
UNUSED_5	5	Unused
PREVIOUS_NODE	6	Previous node (proximate sender)
BUNDLE_AGE	7	Bundle age (in milliseconds)
unassigned	8-9	Unassigned
HOP_COUNT	10	Hop count (#prior xmit attempts)
INTEGRITY	11	Integrity
CONFIDENTIALITY	12	Confidentiality
PRIORITY	13	Priority
RESERVED_MAX_BLOCK_TYPES	14	Reserved max block types
unassigned	15-191	Unassigned
unassigned	>191	Reserved for private and/or experimental use

4.7.3 BPv7 Bundle Status Report Reason Codes

Table 4-22: BPv7 Bundle Status Report Reason Codes defines BPv7 Bundle Status Report Reason Codes enumerated unsigned 64-bit integer values.

Table 4-22: BPv7 Bundle Status Report Reason Codes

Mnemonic	Value	Description
NO_FURTHER_INFORMATION	0	No additional information
LIFETIME_EXPIRED	1	Lifetime expired
FORWARDED_OVER_UNIDIRECTIONAL_LINK	2	Forwarded over unidirectional link
TRANSMISSION_CANCELLED	3	Transmission canceled
DEPLETED_STORAGE	4	Depleted storage
DESTINATION_EID_UNINTELLIGIBLE	5	Destination endpoint ID unavailable
NO_KNOWN_ROUTE_DESTINATION_FROM_HERE	6	No known route to destination from here
NO_TIMELY_CONTACT_WITH_NEXT_NODE_ON_ROUTE	7	No timely contact with next node on route
BLOCK_UNINTELLIGIBLE	8	Block unintelligible
HOP_LIMIT_EXCEEDED	9	Hop limit exceeded
TRAFFIC_PARED	10	Traffic pared (e.g., status reports)
BLOCK_UNSUPPORTED	11	Block unsupported
unassigned	12-254	Unassigned
unassigned	>254	Reserved

4.7.4 BPv7 CRC Type

Table 4-23: BPv7 CRC Type defines BPv7 Cyclic Redundancy Check (CRC) Type enumerated unsigned 8-bit integer values.

Table 4-23: BPv7 CRC Type

Mnemonic	Value	Description
NONE	0	No CRC is present
CRC16_X25	1	A standard X-25 CRC-16 is present
CRC32C	2	A standard CRC32C (Castagnoli) CRC-32 is present
unassigned	3-255	Unassigned

4.7.5 BPv7 Custody Signal Disposition Code

Table 4-24: BPv7 Custody Signal Disposition Code defines BPv7 Custody Signal Disposition Code enumerated unsigned 64-bit integer values.

Table 4-24: BPv7 Custody Signal Disposition Code

Mnemonic	Value	Description
CUSTODY_ACCEPTED	0	Custody accepted
NO_FURTHER_INFORMATION	1	No further information
RESERVED_2	2	Reserved
REDUNDANT	3	Redundant
DEPLETED_STORAGE	4	Depleted storage
DESTINATION_EID_UNINTELLIGIBLE	5	Destination endpoint ID unavailable
NO_KNOWN_ROUTE_DESTINATION_FROM_HERE	6	No known route to destination from here
NO_TIMELY_CONTACT_WITH_NEXT_NODE_ON_ROUTE	7	No timely contact with next node on route
BLOCK_UNINTELLIGIBLE	8	Block unintelligible
unassigned	>8	Unassigned

4.7.6 BPv7 Priority

Table 4-25: BPv7 Priority defines BPv7 Priority enumerated unsigned 64-bit integer values.

Table 4-25: BPv7 Priority

Mnemonic	Value	Description
BULK	0	Bulk priority
NORMAL	1	Normal priority
EXPEDITED	2	Expedited priority
MAX_PRIORITY	2	Maximum priority
DEFAULT	2	Default priority
unassigned	>2	Unassigned

NOTE: The enumerated values defined here are HDTN-specific. CCSDS has not defined BPv7 Bundle Priority, but plans to do so in a future update to the BPv7 standard. When the CCSDS update to BPv7 is published, HDTN definitions for Bundle Priority will need to be re-examined.

4.8 Bundle Storage Enumerations

The following section defines enumerations specific to Storage implementation.

4.8.1 Bundle Storage Duplicate Expiry Order

Table 4-26: Bundle Storage Duplicate Expiry Order defines Bundle Storage Duplicate Expiry Order enumerated 32-bit integer values.

Table 4-26: Bundle Storage Duplicate Expiry Order

Mnemonic	Value	Description
unassigned	<0	Unassigned
FIFO	0	First in, first out
FILO	1	First in, last out
SEQUENCE_NUMBER	2	Sequence number
unassigned	>2	Unassigned

4.9 Concise Binary Object Representation Object Signing and Encryption Enumerations

The following section defines enumerations specific to Concise Binary Object Representation (CBOR) Object Signing and Encryption (COSE) implementation.

4.9.1 COSE Algorithms

Table 4-27: COSE Algorithms defines COSE Algorithms enumerated 32-bit integer values.

Table 4-27: COSE Algorithms

Mnemonic	Value	Description
unassigned	<1	Unassigned
A128GCM	1	AES-GCM mode with 128-bit key, 128-bit tag
unassigned	2	Unassigned
A256GCM	3	AES-GCM mode with 256-bit key, 128-bit tag
unassigned	4	Unassigned
HMAC_256_256	5	Hash Message Authentication Code with SHA-256
HMAC_384_384	6	Hash Message Authentication Code with SHA-384
HMAC_512_512	7	Hash Message Authentication Code with SHA-512
unassigned	>7	Unassigned

4.9.2 COSE Header Parameters

Table 4-28: COSE Header Parameters defines COSE Header Parameters enumerated 32-bit integer values.

Table 4-28: COSE Header Parameters

Mnemonic	Value	Description
unassigned	<1	Unassigned
alg	1	Cryptographic algorithm to use
crit	2	Critical headers to be understood
content_type	3	Content type of the payload
kid	4	Key Identifier (KID)
IV	5	Full initialization vector
Partial_IV	6	Partial initialization vector
counter_signature	7	CBOR-encoded signature structure
unassigned	8	Unassigned
CounterSignature0	9	Counter signature with implied signer and headers
kid_context	10	Identifies the context for the key identifier
unassigned	>10	Unassigned

4.9.3 COSE Key Common Parameters

Table 4-29: COSE Key Common Parameters defines COSE Key Common Parameters enumerated 32-bit integer values.

Table 4-29: COSE Key Common Parameters

Mnemonic	Value	Description
unassigned	<1	Unassigned
kty	1	Identification of the key type
kid	2	Key identification value - match to KID in message
alg	3	Key usage restriction to this algorithm
key_ops	4	Restrict set of permissible operations
Base_IV	5	Base initialization vector to be XORed with partial initialization vectors
unassigned	>5	Unassigned

4.9.4 COSE Key Types

Table 4-30: COSE Key Types defines COSE Key Types enumerated 32-bit integer values.

Table 4-30: COSE Key Types

Mnemonic	Value	Description
unassigned	<1	Unassigned
OKP	1	Octet key pair
EC2	2	Elliptic curve keys with x- and y-coordinate pair
RSA	3	Rivest–Shamir–Adleman key
Symmetric	4	Symmetric keys
HSS_LMS	5	Public key for Hierarchical Signature System/Leighton-Micali Signature hash-based digital signature
WalnutDSA	6	Walnut Digital Signature Algorithm public key
unassigned	>6	Unassigned

4.10 Configuration Enumerations

The following section defines enumerations specific to Configuration implementation.

4.10.1 BPSec Security Context Parameter Name

Table 4-31: BPSec Security Context Parameter Name defines BPSec Security Context Parameter Name enumerated unsigned 8-bit integer values.

Table 4-31: BPSec Security Context Parameter Name

Mnemonic	Value	Description
UNDEFINED	0	Undefined
AES_VARIANT	1	AES variant
SHA_VARIANT	2	SHA variant
IV_SIZE_BYTES	3	IV size bytes
SCOPE_FLAGS	4	Scope flags
SECURITY_BLOCK_CRC	5	Security block CRC
KEY_ENCRYPTION_KEY_FILE	6	Key encryption key file
KEY_FILE	7	Key file
RESERVED_MAX_PARAM_NAMES	8	Reserved max parameter names
unassigned	>8	Unassigned

4.10.2 BPSec Security Context Parameter Type

Table 4-32: BPSec Security Context Parameter Type defines BPSec Security Context Parameter Type enumerated unsigned 8-bit integer values.

Table 4-32: BPSec Security Context Parameter Type

Mnemonic	Value	Description
UNDEFINED	0	Undefined
U64	1	Unsigned 64-bit integer
PATH	2	File path
unassigned	>2	Unassigned

4.10.3 BPSec Security Failure Event

Table 4-33: BPSec Security Failure Event defines BPSec Security Failure Event enumerated unsigned 8-bit integer values.

Table 4-33: BPSec Security Failure Event

Mnemonic	Value	Description
UNDEFINED	0	Undefined
SECURITY_OPERATION_MISCONFIGURED_AT_VERIFIER	1	Security operation misconfigured at verifier
SECURITY_OPERATION_MISSING_AT_VERIFIER	2	Security operation missing at verifier
SECURITY_OPERATION_CORRUPTED_AT_VERIFIER	3	Security operation corrupted at verifier
SECURITY_OPERATION_MISCONFIGURED_AT_ACCEPTOR	4	Security operation misconfigured at acceptor
SECURITY_OPERATION_MISSING_AT_ACCEPTOR	5	Security operation missing at acceptor
SECURITY_OPERATION_CORRUPTED_AT_ACCEPTOR	6	Security operation corrupted at acceptor
RESERVED_MAX_EVENTS	7	Reserved max events
unassigned	>7	Unassigned

4.11 Delay Tolerant Networking RTP Frame Enumerations

The following section defines enumerations specific to DTN RTP Frame implementation.

4.11.1 RTCP Frame Type

Table 4-34: RTCP Frame Type defines Real-Time Transport Control Protocol (RTCP) Frame Type enumerated 32-bit integer values.

Table 4-34: RTCP Frame Type

Mnemonic	Value	Description
unassigned	<200	Unassigned
RTCP_FT_SR	200	Sender report
RTCP_FT_RR	201	Receiver report
RTCP_FT_SDES	202	Source description
RTCP_FT_BYE	203	Goodbye
RTCP_FT_APP	204	Application specific message
unassigned	>204	Unassigned

4.11.2 RTP Error Codes

Table 4-35: RTP Error Codes defines RTP Error Codes enumerated 32-bit integer values.

Table 4-35: RTP Error Codes

Mnemonic	Value	Description
unassigned	<0	Unassigned
RTP_CONCATENATE	0	Concatenate
RTP_INVALID_HEADER	1	Invalid header
RTP_MISMATCH_SSRC	2	Mismatch Synchronization Source Identifier (SSRC)
RTP_INVALID_VERSION	3	Invalid version
RTP_PUSH_PREVIOUS_FRAME	4	Push previous frame
RTP_OUT_OF_SEQ	5	Out of sequence
RTP_FIRST_FRAME	6	First frame
unassigned	>6	Unassigned

4.11.3 RTP Modes

Table 4-36: RTP Modes defines RTP Modes enumerated 32-bit integer values.

Table 4-36: RTP Modes

Mnemonic	Value	Description
unassigned	<1	Unassigned
RTP_RECV_ONLY	1	Receive only
RTP_SEND_ONLY	2	Send only
RTP_SEND_RECV	3	Send and receive
unassigned	>3	Unassigned

4.12 Delay Tolerant Networking Utility Enumerations

The following section defines enumerations specific to DTN Utility implementation.

4.12.1 RTP Format

Table 4-37: RTP Format defines RTP Format enumerated 32-bit integer values.

Table 4-37: RTP Format

Mnemonic	Value	Description
unassigned	<0	Unassigned
RTP_FORMAT_GENERIC	0	Generic
RTP_FORMAT_PCMU	0	Pulse Code Modulation (PCM) μ-law, International Telecommunication Union Telecommunication Standardization Sector (ITU-T) G.711 audio
reserved	1-2	Reserved
RTP_FORMAT_GSM	3	European Global System for Mobile Communications Full Rate audio 13 kbit/s (GSM 06.10)
RTP_FORMAT_G723	4	ITU-T G.723.1 audio
RTP_FORMAT_DVI4_32	5	Interactive Multimedia Association (IMA) Adaptive Differential Pulse Code Modulation (ADPCM) audio 32 kbit/s
RTP_FORMAT_DVI4_64	6	IMA ADPCM audio 64 kbit/s
RTP_FORMAT_LPC	7	Experimental Linear Predictive Coding (LPC) audio 5.6 kbit/s
RTP_FORMAT_PCMA	8	ITU-T G.711 PCM A-Law audio 64 kbit/s
RTP_FORMAT_G722	9	ITU-T G.722 audio 64 kbit/s
RTP_FORMAT_L16_STEREO	10	Linear PCM 16-bit stereo audio 1411.2 kbit/s, uncompressed
RTP_FORMAT_L16_MONO	11	Linear PCM 16-bit audio 705.6 kbit/s, uncompressed
unassigned	12	Qualcomm code excited linear prediction (unsupported)
unassigned	13	Comfort noise (unsupported)
unassigned	14	Moving Picture Experts Group (MPEG)-1 or MPEG-2 audio only (unsupported)
RTP_FORMAT_G728	15	ITU-T G.728 audio 16 kbit/s
RTP_FORMAT_DVI4_441	16	IMA ADPCM audio 44.1 kbit/s
RTP_FORMAT_DVI4_882	17	IMA ADPCM audio 88.2 kbit/s
RTP_FORMAT_G729	18	ITU-T G.729 and G.729a audio 8 kbit/s
reserved	19	Reserved
unassigned	20-24	Unassigned
unassigned	25	Sun CellB video (unsupported)
unassigned	26	Joint Photographic Experts Group (JPEG) video (unsupported)
unassigned	27	Unassigned
unassigned	28	Xerox Palo Alto Research Center (PARC) network video (unsupported)
unassigned	29-30	Unassigned
unassigned	31	ITU-T H.261 video (unsupported)
unassigned	32	MPEG-1 and MPEG-2 video (unsupported)
unassigned	33	MPEG-2 transport stream (unsupported)
unassigned	34	H.263 video, first version (1996) (unsupported)
unassigned	35-71	Unassigned
reserved	72-76	Reserved
unassigned	77-95	Unassigned
RTP_FORMAT_DYNAMICRTP	96	Dynamic RTP, Fast Forward MPEG (FFMPEG)
RTP_FORMAT_G726_32	97	ITU-T G.726 audio 32 kbit/s
RTP_FORMAT_G726_24	98	ITU-T G.726 audio 24 kbit/s
RTP_FORMAT_G726_16	99	ITU-T G.726 audio 16 kbit/s
RTP_FORMAT_G729D	100	ITU-T G.729 Annex D
RTP_FORMAT_G729E	101	ITU-T G.729 Annex E

Mnemonic	Value	Description
RTP_FORMAT_GSM_EFR	102	ITU-T GSM Enhanced Full-Rate (GRM-EFR) Speech Transcoding (GSM 06.60)
RTP_FORMAT_L8	103	Linear PCM 8-bit audio with 128 offset
RTP_FORMAT_VDVI	104	Variable-Rate DVI4 audio
RTP_FORMAT_OPUS	105	Opus audio
RTP_FORMAT_H264	106	H.264 video (MPEG-4 Part 10) Advanced Video Coding (AVC)
RTP_FORMAT_H265	107	H.265 video High Efficiency Video Coding (HEVC)
RTP_FORMAT_H266	108	H.266 Versatile Video Coding (VVC) (MPEG-I Part 3) (unsupported)
unassigned	>108	Unassigned

4.13 Licklider Transmission Protocol Enumerations

The following section defines enumerations specific to Licklider Transmission Protocol (LTP) implementation.

4.13.1 Cancel Segment Reason Codes

Table 4-38: Cancel Segment Reason Codes defines Cancel Segment Reason Codes enumerated 32-bit integer values.

Table 4-38: Cancel Segment Reason Codes

Mnemonic	Value	Description
unassigned	<0	Unassigned
USER_CANCELLED	0	Client service canceled session
UNREACHABLE	1	Unreachable client service
RLEXC	2	Retransmission limit exceeded
MISCOLORED	3	Received either: • a red-part data segment at block offset above any green-part data segment offset • a green-part data segment at block offset below any red-part data segment offset
SYSTEM_CANCELLED	4	A system error condition caused unexpected session termination
RXMTCYCEXC	5	Exceeded the retransmission-cycles limit
RESERVED	6	Reserved
Unassigned	>6	Unassigned

4.13.2 LTP Data Segment Receiver State

Table 4-39: LTP Data Segment Receiver State defines LTP Data Segment Receiver State enumerated 32-bit integer values.

Table 4-39: LTP Data Segment Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_CLIENT_SERVICE_ID_SDNV	0	Read client service ID self-delimiting numeric value
READ_OFFSET_SDNV	1	Read offset self-delimiting numeric value
READ_LENGTH_SDNV	2	Read length self-delimiting numeric value
READ_CHECKPOINT_SERIAL_NUMBER_SDNV	2	Read checkpoint serial number self-delimiting numeric value
READ_REPORT_SERIAL_NUMBER_SDNV	4	Read report serial number self-delimiting numeric value
READ_CLIENT_SERVICE_DATA	5	Read client service data
Unassigned	>5	Unassigned

4.13.3 LTP Data Segment Type Flags

Table 4-40: LTP Data Segment Type Flags defines LTP Data Segment Type Flags enumerated 32-bit integer values.

Table 4-40: LTP Data Segment Type Flags

Mnemonic	Value	Description
unassigned	<0	Unassigned
REDDATA	0	Reliably transmitted data
REDDATA_CHECKPOINT	1	Reliably transmitted data, checkpoint
REDDATA_CHECKPOINT_ENDOFREDPART	2	Reliably transmitted data, end of reliably transmitted data part
REDDATA_CHECKPOINT_ENDOFREDPART_ENDOFBLOCK	3	Reliably transmitted data, end of block
GREENDATA	4	Unreliably transmitted data
Unassigned	5-6	Undefined
GREENDATA_ENDOFBLOCK	7	Unreliably transmitted data, end of block
Unassigned	>7	Unassigned

4.13.4 LTP Header Receiver State

Table 4-41: LTP Header Receiver State defines LTP Header Receiver State enumerated 32-bit integer values.

Table 4-41: LTP Header Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_CONTROL_BYTE	0	Read control byte
READ_SESSION_ORIGINATOR_ENGINE_ID_SDNV	1	Read session originator engine ID self-delimiting numeric value
READ_SESSION_NUMBER_SDNV	2	Read session number self-delimiting numeric value
READ_NUM_EXTENSIONS_BYTE	3	Read number of extensions byte
READ_ONE_HEADER_EXTENSION_TAG_BYTE	4	Read one header extension tag byte
READ_ONE_HEADER_EXTENSION_LENGTH_SDNV	5	Read one header extension length self-delimiting numeric value
READ_ONE_HEADER_EXTENSION_VALUE	6	Read one header extension value
Unassigned	>6	Unassigned

4.13.5 LTP Main Receiver State

Table 4-42: LTP Main Receiver State defines LTP Main Receiver State enumerated 32-bit integer values.

Table 4-42: LTP Main Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_HEADER	0	Read header
READ_DATA_SEGMENT_CONTENT	1	Read data segment content
READ_REPORT_SEGMENT_CONTENT	2	Read report segment content
READ_REPORT_ACKNOWLEDGEMENT_SEGMENT_CONTENT	3	Read report acknowledgement segment content
READ_CANCEL_SEGMENT_CONTENT_BYTE	4	Read cancel segment content byte
READ_TRAILER	5	Read trailer
Unassigned	>5	Unassigned

4.13.6 LTP Trailer Receiver State

Table 4-43: LTP Trailer Receiver State defines LTP Trailer Receiver State enumerated 32-bit integer values.

Table 4-43: LTP Trailer Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_ONE_TRAILER_EXTENSION_TAG_BYTE	0	Read one header extension tag byte
READ_ONE_TRAILER_EXTENSION_LENGTH_SDNV	1	Read one header extension length self-delimiting numeric value
READ_ONE_TRAILER_EXTENSION_VALUE	2	Read one header extension value
Unassigned	>2	Unassigned

4.13.7 LTP Report Acknowledgement Segment Receiver State

Table 4-44: LTP Report Acknowledgement Segment Receiver State defines LTP Report Acknowledgement Segment Receiver State enumerated 32-bit integer values.

Table 4-44: LTP Report Acknowledgement Segment Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_REPORT_SERIAL_NUMBER_SDNV	0	Read report serial number self-delimiting numeric value
Unassigned	>0	Unassigned

4.13.8 LTP Report Segment Receiver State

Table 4-45: LTP Report Segment Receiver State defines LTP Report Segment Receiver State enumerated 32-bit integer values.

Table 4-45: LTP Report Segment Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_REPORT_SERIAL_NUMBER_SDNV	0	Read report serial number self-delimiting numeric value
READ_CHECKPOINT_SERIAL_NUMBER_SDNV	1	Read checkpoint serial number self-delimiting numeric value
READ_UPPER_BOUND_SDNV	2	Read upper bound self-delimiting numeric value
READ_LOWER_BOUND_SDNV	2	Read lower bound self-delimiting numeric value
READ_RECEPTION_CLAIM_COUNT_SDNV	4	Read reception claim count self-delimiting numeric value
READ_ONE_RECEPTION_CLAIM_OFFSET_SDNV	5	Read one reception claim offset self-delimiting numeric value
READ_ONE_RECEPTION_CLAIM_LENGTH_SDNV	6	Read one reception claim length self-delimiting numeric value
Unassigned	>6	Unassigned

4.13.9 LTP Segment Type Flags

Table 4-46: LTP Segment Type Flags defines LTP Segment Type Flags enumerated 32-bit integer values.

Table 4-46: LTP Segment Type Flags

Mnemonic	Value	Description
unassigned	<0	Unassigned
REDDATA	0	Reliably transmitted data
REDDATA_CHECKPOINT	1	Reliably transmitted data, checkpoint
REDDATA_CHECKPOINT_ENDOFREDPART	2	Reliably transmitted data, end of reliably transmitted data part
REDDATA_CHECKPOINT_ENDOFREDPART_ENDOFBLOCK	3	Reliably transmitted data, end of block
GREENDATA	4	Unreliably transmitted data
Unassigned	5-6	Undefined
GREENDATA_ENDOFBLOCK	7	Unreliably transmitted data, end of block
REPORT_SEGMENT	8	Report segment
REPORT_ACK_SEGMENT	9	Report acknowledgement segment
Unassigned	10-11	Undefined
CANCEL_SEGMENT_FROM_BLOCK_SENDER	12	Cancel segment from block sender
CANCEL_ACK_SEGMENT_TO_BLOCK_SENDER	13	Cancel acknowledgment segment.to block sender
CANCEL_SEGMENT_FROM_BLOCK_RECEIVER	14	Cancel segment from block receiver
CANCEL_ACK_SEGMENT_TO_BLOCK_RECEIVER	15	Cancel acknowledgment segment to block receiver
Unassigned	>15	Unassigned

4.14 Logger Enumerations

The following section defines enumerations specific to Logger implementation.

4.14.1 Process

Table 4-47: Process defines Process enumerated 32-bit integer values.

Table 4-47: Process

Mnemonic	Value	Description
unassigned	<0	Unassigned
Bpgen	0	Bundle protocol generator
Bping	1	Bundle protocol ping
Bpreceivefile	2	Bundle protocol receive file
Bpsendfile	3	Bundle protocol send file
Bpsink	4	Bundle protocol sink
Ltpfiletransfer	5	Licklider Transmission Protocol file transfer
Egress	6	Egress
Telem	7	Telemetry
Hdtnoneprocess	8	HDTN one process
Ingress	9	Ingress
Router	10	Router
Storage	11	Storage
Releasemessagesender	12	Release message sender
Storagespeedtest	13	Storage speed test
Udpdelaysim	14	UDP delay simulator
Unittest	15	Unit test
None	16	None
Unassigned	>16	Unassigned

4.14.2 Subprocess

Table 4-48: Subprocess defines Subprocess enumerated 32-bit integer values.

Table 4-48: Subprocess

Mnemonic	Value	Description
unassigned	<0	Unassigned
Egress	0	Egress
Ingress	1	Ingress
Router	2	Router
Storage	3	Storage
Telem	4	Telemetry
Gui	5	Graphical User Interface (GUI)
Unittest	6	Unit Test
Cli	7	Command Line Interface (CLI)
none	8	None
unassigned	>8	Unassigned

4.15 Message Enumerations

The following section defines enumerations specific to Message implementation.

4.15.1 Egress Acknowledge Error Type

Table 4-49: Egress Acknowledge Error Type defines Egress Acknowledge Error Type enumerated unsigned 8-bit integer values.

Table 4-49: Egress Acknowledge Error Type

Mnemonic	Value	Description
NO_ERRORS	0	No errors
LINK_DOWN	1	Link down
NO_OUTDUCT	2	No outduct
unassigned	>2	Unassigned

4.16 Telemetry Enumerations

The following section defines enumerations specific to Telemetry implementation.

4.16.1 Telemetry API Source

Table 4-50: Telemetry API Source defines Telemetry Application Programming Interface (API) Source enumerated 32-bit integer values.

Table 4-50: Telemetry API Source

Mnemonic	Value	Description
unassigned	<0	Unassigned
Webgui	0	Web GUI
Socket	1	Socket
Unassigned	>1	Unassigned

4.17 Transmission Control Protocol Convergence Layer Enumerations

The following section defines enumerations specific to Transmission Control Protocol (TCP) Convergence Layer (TCPCL) implementation.

4.17.1 TCPCL Bundle Refusal Codes

Table 4-51: TCPCL Bundle Refusal Codes defines TCPCL Bundle Refusal Codes enumerated 32-bit integer values.

Table 4-51: TCPCL Bundle Refusal Codes

Mnemonic	Value	Description
unassigned	<0	Unassigned
REFUSAL_REASON_UNKNOWN	0	Reason unknown
RECEIVER_HAS_COMPLETE_BUNDLE	1	Receiver has complete bundle
RECEIVER_RESOURCES_EXHAUSTED	2	Receiver resources exhausted
RECEIVER_PROBLEM__PLEASE_RETRANSMIT	3	Receiver problem: please retransmit
UNASSIGNED	4	Unassigned
Unassigned	>4	Unassigned

4.17.2 TCPCL Contact Header Receiver State

Table 4-52: TCPCL Contact Header Receiver State defines TCPCL Contact Header Receiver State enumerated 32-bit integer values.

Table 4-52: TCPCL Contact Header Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_SYNC_1	0	Read sync 1
READ_SYNC_2	1	Read sync 2
READ_SYNC_3	2	Read sync 3
READ_SYNC_4	3	Read sync 4
READ_VERSION	4	Read version
READ_FLAGS	5	Read flags
READ_KEEPALIVE_INTERVAL_BYTE1	6	Read keep-alive interval byte 1
READ_KEEPALIVE_INTERVAL_BYTE2	7	Read keep-alive interval byte 2
READ_LOCAL_EID_LENGTH_SDNV	8	Read local endpoint ID length self-delimiting numeric value
READ_LOCAL_EID_STRING	9	Read local endpoint ID string
Unassigned	>9	Unassigned

4.17.3 TCPCL Data Segment Receiver State

Table 4-53: TCPCL Data Segment Receiver State defines TCPCL Data Segment Receiver State enumerated 32-bit integer values.

Table 4-53: TCPCL Data Segment Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_CONTENT_LENGTH_SDNV	0	Read content length self-delimiting numeric value
READ_CONTENTS	1	Read contents
Unassigned	>1	Unassigned

4.17.4 TCPCL Main Receiver State

Table 4-54: TCPCL Main Receiver State defines TCPCL Main Receiver State enumerated 32-bit integer values.

Table 4-54: TCPCL Main Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_CONTACT_HEADER	0	Read contact header
READ_MESSAGE_TYPE_BYTE	1	Read message type byte
READ_DATA_SEGMENT	2	Read data segment
READ_ACK_SEGMENT	3	Read acknowledge segment
READ_LENGTH_SEGMENT	4	Read length segment
READ_SHUTDOWN_SEGMENT_REASON_CODE	5	Read shutdown segment reason code
READ_SHUTDOWN_SEGMENT_RECONNECTION_DELAY_SDNV	6	Read shutdown segment reconnection delay self-delimiting numeric value
Unassigned	>6	Unassigned

4.17.5 TCPCL Message Type Byte Codes

Table 4-55: TCPCL Message Type Byte Codes defines TCPCL Message Type Byte Codes enumerated 32-bit integer values.

Table 4-55: TCPCL Message Type Byte Codes

Mnemonic	Value	Description
unassigned	<0	Unassigned
RESERVED	0	Reserved
DATA_SEGMENT	1	Data segment
ACK_SEGMENT	2	Acknowledgement segment
REFUSE_BUNDLE	3	Refuse bundle
KEEPALIVE	4	Keep-alive
SHUTDOWN	5	Shutdown
LENGTH	6	Length
Unassigned	>6	Unassigned

4.17.6 TCPCL Shutdown Reason Codes

Table 4-56: TCPCL Shutdown Reason Codes defines TCPCL Shutdown Reason Codes enumerated 32-bit integer values.

Table 4-56: TCPCL Shutdown Reason Codes

Mnemonic	Value	Description
unassigned	<0	Unassigned
IDLE_TIMEOUT	0	Idle timeout
VERSION_MISMATCH	1	Version mismatch
BUSY	2	Busy
UNASSIGNED	3	Unassigned
Unassigned	>3	Unassigned

4.18 Transmission Control Protocol Convergence Layer Version 4 Enumerations

The following section defines enumerations specific to TCPCL version 4 (TCPCLv4) implementation.

4.18.1 TCPCLv4 Contact Header Receiver State

Table 4-57: TCPCLv4 Contact Header Receiver State defines TCPCLv4 Contact Header Receiver State enumerated 32-bit integer values.

Table 4-57: TCPCLv4 Contact Header Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_SYNC_1	0	Read sync 1
READ_SYNC_2	1	Read sync 2
READ_SYNC_3	2	Read sync 3
READ_SYNC_4	3	Read sync 4
READ_VERSION	4	Read version
READ_FLAGS	5	Read flags
READ_KEEPALIVE_INTERVAL_BYTE1	6	Read keep-alive interval byte 1
READ_KEEPALIVE_INTERVAL_BYTE2	7	Read keep-alive interval byte 2
READ_LOCAL_EID_LENGTH_SDNV	8	Read local endpoint ID length self-delimiting numeric value
READ_LOCAL_EID_STRING	9	Read local endpoint ID string
Unassigned	>9	Unassigned

4.18.2 TCPCLv4 Data Acknowledgement Receiver State

Table 4-58: TCPCLv4 Data Acknowledgement Receiver State defines TCPCLv4 Data Acknowledgement Receiver State enumerated 32-bit integer values.

Table 4-58: TCPCLv4 Data Acknowledgement Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_MESSAGE_FLAGS_BYTE	0	Read message flags byte
READ_TRANSFER_ID_U64	1	Read transfer ID uint64
READ_ACKNOWLEDGED_LENGTH_U64	2	Read acknowledged length uint64
unassigned	>2	Unassigned

4.18.3 TCPCLv4 Data Segment Receiver State

Table 4-59: TCPCLv4 Data Segment Receiver State defines TCPCLv4 Data Segment Receiver State enumerated 32-bit integer values.

Table 4-59: TCPCLv4 Data Segment Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_MESSAGE_FLAGS_BYTE	0	Read message flags byte
READ_TRANSFER_ID_U64	1	Read transfer ID uint64
READ_START_SEGMENT_TRANSFER_EXTENSION_ITEMS_LENGTH_U32	2	Read start segment transfer extension items length uint32
READ_ONE_START_SEGMENT_TRANSFER_EXTENSION_ITEM_FLAG	3	Read one start segment transfer extension item flag
READ_ONE_START_SEGMENT_TRANSFER_EXTENSION_ITEM_TYPE	4	Read one start segment transfer extension item type

Mnemonic	Value	Description
READ_ONE_START_SEGMENT_TRANSFER_EXTENSION_ITEM_LENGTH	5	Read one start segment transfer extension item length
READ_ONE_START_SEGMENT_TRANSFER_EXTENSION_ITEM_VALUE	6	Read one start segment transfer extension item value
READ_DATA_LENGTH_U64	7	Read data length uint64
READ_DATA_CONTENTS	8	Read data contents
Unassigned	>8	Unassigned

4.18.4 TCPCLv4 Main Receiver State

Table 4-60: TCPCLv4 Main Receiver State defines TCPCLv4 Main Receiver State enumerated 32-bit integer values.

Table 4-60: TCPCLv4 Main Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_CONTACT_HEADER	0	Read contact header
READ_MESSAGE_TYPE_BYTE	1	Read message type byte
READ_DATA_SEGMENT	2	Read data segment
READ_ACK_SEGMENT	3	Read acknowledge segment
READ_TRANSFER_REFUSAL	4	Read transfer refusal
READ_MESSAGE_REJECTION	5	Read message rejection
READ_LENGTH_SEGMENT	6	Read length segment
READ_SESSION_TERMINATION	7	Read session termination
READ_SESSION_INIT	8	Read session initialization
Unassigned	>8	Unassigned

4.18.5 TCPCLv4 Message Rejection Reason Codes

Table 4-61: TCPCLv4 Message Rejection Reason Codes defines TCPCLv4 Message Rejection Reason Codes enumerated 32-bit integer values.

Table 4-61: TCPCLv4 Message Rejection Reason Codes

Mnemonic	Value	Description
unassigned	<0	Unassigned
MESSAGE_TYPE_UNKNOWN	0	Message type code in received message is not recognized
MESSAGE_UNSUPPORTED	1	Message type is not supported by the TCPCL entity
MESSAGE_UNEXPECTED	2	Message type is not expected in the current session state
Unassigned	>2	Unassigned

4.18.6 TCPCLv4 Message Rejection Receiver State

Table 4-62: TCPCLv4 Message Rejection Receiver State defines TCPCLv4 Message Rejection Receiver State enumerated 32-bit integer values.

Table 4-62: TCPCLv4 Message Rejection Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_REASON_CODE_BYTE	0	Read reason code byte
READ_REJECTED_MESSAGE_HEADER	1	Read rejected message header
Unassigned	>1	Unassigned

4.18.7 TCPCLv4 Message Type Byte Codes

Table 4-63: TCPCLv4 Message Type Byte Codes defines TCPCLv4 Message Type Byte Codes enumerated 32-bit integer values.

Table 4-63: TCPCLv4 Message Type Byte Codes

Mnemonic	Value	Description
unassigned	<0	Unassigned
RESERVED	0	Reserved
XFER_SEGMENT	1	Transfer segment
XFER_ACK	2	Transfer acknowledgement
XFER_REFUSE	3	Transfer refusal
KEEPALIVE	4	Keep-alive
SESS_TERM	5	Session termination
MSG_REJECT	6	Message rejection
SESS_INIT	7	Session initiation
Unassigned	>7	Unassigned

4.18.8 TCPCLv4 Session Initialization Receiver State

Table 4-64: TCPCLv4 Session Initialization Receiver State defines TCPCLv4 Session Termination Receiver State enumerated 32-bit integer values.

Table 4-64: TCPCLv4 Session Initialization Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_KEEPALIVE_INTERVAL_U16	0	Read keep-alive interval uint16
READ_SEGMENT_MRU_U64	1	Read segment maximum receive unit uint64
READ_TRANSFER_MRU_U64	2	Read transfer maximum receive unit uint64
READ_NODE_ID_LENGTH_U16	3	Read node ID length uint16
READ_NODE_ID	4	Read node ID
READ_SESSION_EXTENSION_ITEMS_LENGTH_U32	5	Read session extension items length uint32
READ_ONE_SESSION_EXTENSION_ITEM_FLAG	6	Read one session extension item flag
READ_ONE_SESSION_EXTENSION_ITEM_TYPE	7	Read one session extension item type
READ_ONE_SESSION_EXTENSION_ITEM_LENGTH	8	Read one session extension item length
READ_ONE_SESSION_EXTENSION_ITEM_VALUE	9	Read one session extension item value
Unassigned	>9	Unassigned

4.18.9 TCPCLv4 Session Termination Reason Codes

Table 4-65: TCPCLv4 Session Termination Reason Codes defines TCPCLv4 Session Termination Reason Codes enumerated 32-bit integer values.

Table 4-65: TCPCLv4 Session Termination Reason Codes

Mnemonic	Value	Description
unassigned	<0	Unassigned
UNKNOWN	0	No reason for termination available
IDLE_TIMEOUT	1	Terminated due to idle connection timeout
VERSION_MISMATCH	2	Incorrect TCPCL protocol version
BUSY	3	Too busy to handle the session
CONTACT_FAILURE	4	Cannot interpret or negotiate a contact header or session initiation option
RESOURCE_EXHAUSTION	5	Resource limits prevent continuing the session
Unassigned	>5	Unassigned

4.18.10 TCPCLv4 Session Termination Receiver State

Table 4-66: TCPCLv4 Session Termination Receiver State defines TCPCLv4 Session Termination Receiver State enumerated 32-bit integer values.

Table 4-66: TCPCLv4 Session Termination Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_MESSAGE_FLAGS_BYTE	0	Read message flags byte
READ_REASON_CODE_BYTE	1	Read reason code byte
Unassigned	>1	Unassigned

4.18.11 TCPCLv4 Transfer Refusal Reason Codes

Table 4-67: TCPCLv4 Transfer Refusal Reason Codes defines TCPCLv4 Transfer Refusal Reason Codes enumerated 32-bit integer values.

Table 4-67: TCPCLv4 Transfer Refusal Reason Codes

Mnemonic	Value	Description
unassigned	<0	Unassigned
REFUSAL_REASON_UNKNOWN	0	Reason for refusal is unknown or not specified
REFUSAL_REASON_ALREADY_COMPLETED	1	Receiver already has the complete bundle
REFUSAL_REASON_NO_RESOURCES	2	Receiver resources are exhausted
REFUSAL_REASON_RETRANSMIT	3	Receiver encountered a problem requiring bundle retransmission
REFUSAL_REASON_NOT_ACCEPTABLE	4	Bundle data or transfer extension data issue was encountered
REFUSAL_REASON_EXTENSION_FAILURE	5	Failure processing transfer extension items
REFUSAL_REASON_SESSION_TERMINATING	6	Receiving entity is terminating the session
Unassigned	>6	Unassigned

4.18.12 TCPCLv4 Transfer Refusal Receiver State

Table 4-68: TCPCLv4 Transfer Refusal Receiver State defines TCPCLv4 Transfer Refusal Receiver State enumerated 32-bit integer values.

Table 4-68: TCPCLv4 Transfer Refusal Receiver State

Mnemonic	Value	Description
unassigned	<0	Unassigned
READ_REASON_CODE_BYTE	0	Read reason code byte
READ_TRANSFER_ID	1	Read transfer ID
unassigned	>1	Unassigned

4.19 Utility Enumerations

The following section defines enumerations specific to Utility implementation.

4.19.1 Directory Monitor Event Type

Table 4-69: Directory Monitor Event Type defines Directory Monitory Event Type enumerated 32-bit integer values.

Table 4-69: Directory Monitor Event Type

Mnemonic	Value	Description
unassigned	<0	Unassigned
null	0	Null
added	1	Added
removed	2	Removed
modified	3	Modified
renamed_old_name	4	Renamed old name
renamed_new_name	5	Renamed new name
recursive_rescan	6	Recursive rescan
unassigned	>6	Unassigned

4.19.2 Encapsulation Packet Type

Table 4-70: Encapsulation Packet Type defines Encapsulation Packet Type enumerated unsigned 8-bit integer values.

Table 4-70: Encapsulation Packet Type

Mnemonic	Value	Description
unassigned	0-223	Unassigned
IDLE	224 (0xE0)	Idle packet
unassigned	225-227	Unassigned
LTP	228 (0xE4)	LTP packet
unassigned	229-239	Unassigned
BP	240 (0xF0)	BP packet
unassigned	>240	Unassigned

4.20 ZeroMQ Enumerations

The following section defines enumerations specific to ZeroMQ implementation.

4.20.1 ZeroMQ Context Options

Definition of the ZeroMQ Context Options enumeration is dependent on definition of the ZMQ_CPP11, ZMQ_BLOCKY, ZMQ_IO_THREADS, ZMQ_THREAD_SCHED_POLICY, ZMQ_THREAD_PRIORITY, ZMQ_THREAD_AFFINITY_CPU_ADD, ZMQ_THREAD_AFFINITY_CPU_REMOVE, ZMQ_THREAD_NAME_PREFIX, ZMQ_MAX_MSGSZ, ZMQ_ZERO_COPY_RECV, ZMQ_MAX_SOCKETS, ZMQ_SOCKET_LIMIT, ZMQ_IPV6, and ZMQ_MSG_T_SIZE symbolic constants.

Table 4-71: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_IO_THREADS Defined) defines ZeroMQ Context Options enumerated 32-bit integer values when ZMQ_CPP11 and ZMQ_IO_THREADS are defined.

Table 4-71: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_IO_THREADS Defined)

Mnemonic	Value	Description
io_threads	ZMQ_IO_THREADS (1)	ZeroMQ context: I/O threads

Table 4-72: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_MAX_SOCKETS Defined) defines ZeroMQ Context Options enumerated 32-bit integer values when ZMQ_CPP11 and ZMQ_MAX_SOCKETS are defined.

Table 4-72: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_MAX_SOCKETS Defined)

Mnemonic	Value	Description
max_sockets	ZMQ_MAX_SOCKETS (2)	ZeroMQ context: maximum sockets

Table 4-73: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_SOCKET_LIMIT Defined) defines ZeroMQ Context Options enumerated 32-bit integer values when ZMQ_CPP11 and ZMQ_SOCKET_LIMIT are defined.

Table 4-73: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_SOCKET_LIMIT Defined)

Mnemonic	Value	Description
socket_limit	ZMQ_SOCKET_LIMIT (3)	ZeroMQ context: socket limit

Table 4-74: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_THREAD_PRIORITY Defined) defines ZeroMQ Context Options enumerated 32-bit integer values when ZMQ_CPP11 and ZMQ_THREAD_PRIORITY are defined.

Table 4-74: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_THREAD_PRIORITY Defined)

Mnemonic	Value	Description
thread_priority	ZMQ_THREAD_PRIORITY (3)	ZeroMQ context: thread priority

Table 4-75: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_THREAD_SCHED_POLICY Defined) defines ZeroMQ Context Options enumerated 32-bit integer values when ZMQ_CPP11 and ZMQ_THREAD_SCHED_POLICY are defined.

Table 4-75: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_THREAD_SCHED_POLICY Defined)

Mnemonic	Value	Description
thread_sched_policy	ZMQ_THREAD_SCHED_POLICY (4)	ZeroMQ context: thread schedule policy

Table 4-76: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_BLOCKY Defined) defines ZeroMQ Context Options enumerated 32-bit integer values when ZMQ_CPP11 and ZMQ_MAX_MSGSZ are defined.

Table 4-76: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_BLOCKY Defined)

Mnemonic	Value	Description
max_msgsz	ZMQ_MAX_MSGSZ (5)	ZeroMQ socket option: maximum message size

Table 4-77: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_MSG_T_SIZE Defined) defines ZeroMQ Context Options enumerated 32-bit integer values when ZMQ_CPP11 and ZMQ_MSG_T_SIZE are defined.

Table 4-77: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_MSG_T_SIZE Defined)

Mnemonic	Value	Description
msg_t_size	ZMQ_MSG_T_SIZE (6)	ZeroMQ context: message structure size

Table 4-78: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_THREAD_AFFINITY_CPU_ADD Defined) defines ZeroMQ Context Options enumerated 32-bit integer values when ZMQ_CPP11 and ZMQ_THREAD_AFFINITY_CPU_ADD are defined.

Table 4-78: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_THREAD_AFFINITY_CPU_ADD Defined)

Mnemonic	Value	Description
thread_affinity_cpu_add	ZMQ_THREAD_AFFINITY_CPU_ADD (7)	ZeroMQ context: thread affinity, add CPU

Table 4-79: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_THREAD_AFFINITY_CPU_REMOVE Defined) defines ZeroMQ Context Options enumerated 32-bit integer values when ZMQ_CPP11 and ZMQ_THREAD_AFFINITY_CPU_REMOVE are defined.

Table 4-79: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_THREAD_AFFINITY_CPU_REMOVE Defined)

Mnemonic	Value	Description
thread_affinity_cpu_remove	ZMQ_THREAD_AFFINITY_CPU_REMOVE (8)	ZeroMQ context: thread affinity, remove CPU

Table 4-80: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_THREAD_NAME_PREFIX Defined) defines ZeroMQ Context Options enumerated 32-bit integer values when ZMQ_CPP11 and ZMQ_THREAD_NAME_PREFIX are defined.

Table 4-80: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_THREAD_NAME_PREFIX Defined)

Mnemonic	Value	Description
thread_name_prefix	ZMQ_THREAD_NAME_PREFIX (9)	ZeroMQ context: thread name prefix

Table 4-81: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_ZERO_COPY_RECV Defined) defines ZeroMQ Context Options enumerated 32-bit integer values when ZMQ_CPP11 and ZMQ_ZERO_COPY_RECV are defined.

Table 4-81: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_ZERO_COPY_RECV Defined)

Mnemonic	Value	Description
zero_copy_recv	ZMQ_ZERO_COPY_RECV (10)	ZeroMQ context: zero copy receive

Table 4-82: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_IPV6 Defined) defines ZeroMQ Context Options enumerated 32-bit integer values when ZMQ_CPP11 and ZMQ_IPV6 are defined.

Table 4-82: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_IPV6 Defined)

Mnemonic	Value	Description
ipv6	ZMQ_IPV6 (42)	ZeroMQ socket option: Ipv6

Table 4-83: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_BLOCKY Defined) defines ZeroMQ Context Options enumerated 32-bit integer values when ZMQ_CPP11 and ZMQ_BLOCKY are defined.

Table 4-83: ZeroMQ Context Options (ZMQ_CPP11 Defined and ZMQ_BLOCKY Defined)

Mnemonic	Value	Description
blocky	ZMQ_BLOCKY (70)	ZeroMQ socket option: blocky

4.20.2 ZeroMQ Event Flags

Definition of the ZeroMQ Event Flags enumeration is dependent on definition of the ZMQ_BUILD_DRAFT_API, ZMQ_CPP11, and ZMQ_HAVE_POLLER symbolic constants.

Table 4-84: ZeroMQ Event Flags (ZMQ_BUILD_DRAFT_API Defined AND ZMQ_CPP11 Defined AND ZMQ_HAVE_POLLER Defined) defines ZeroMQ Event Flags enumerated 16-bit integer values when ZMQ_BUILD_DRAFT_API, ZMQ_CPP11, and ZMQ_HAVE_POLLER are defined.

Table 4-84: ZeroMQ Event Flags (ZMQ_BUILD_DRAFT_API Defined AND ZMQ_CPP11 Defined ANDZMQ_HAVE_POLLER Defined)

Mnemonic	Value	Description
unassigned	<0	Unassigned
None	0	None
Pollin	ZMQ_POLLIN (1)	Data may be read from the socket
pollout	ZMQ_POLLOUT (2)	Data may be written to the socket
unassigned	3	Unassigned
Pollerr	ZMQ_POLLERR (4)	Error condition is present on the socket
unassigned	5-7	Unassigned
Pollpri	ZMQ_POLLPRI (8)	Urgent data is available to read from the socket
unassigned	>8	Unassigned

4.20.3 ZeroMQ Receive Flags

Definition of the ZeroMQ Receive Flags enumeration is dependent on definition of the ZMQ_CPP11 symbolic constant.

Table 4-85: ZeroMQ Receive Flags (ZMQ_CPP11 Defined) defines ZeroMQ Receive Flags enumerated integer values when ZMQ_CPP11 is defined.

Table 4-85: ZeroMQ Receive Flags (ZMQ_CPP11 Defined)

Mnemonic	Value	Description
unassigned	<0	Unassigned
None	0	None
Dontwait	ZMQ_DONTWAIT (1)	Send/receive option: don't wait
unassigned	>1	Unassigned

4.20.4 ZeroMQ Send Flags

Definition of the ZeroMQ Send Flags enumeration is dependent on definition of the ZMQ_CPP11 symbolic constant.

Table 4-86: ZeroMQ Send Flags (ZMQ_CPP11 Defined) defines ZeroMQ Send Flags enumerated integer values when ZMQ_CPP11 is defined.

Table 4-86: ZeroMQ Send Flags (ZMQ_CPP11 Defined)

Mnemonic	Value	Description
unassigned	<0	Unassigned
None	0	None
Dontwait	ZMQ_DONTWAIT (1)	Send/receive option: don't wait
sndmore	ZMQ_SNDMORE (2)	Send/receive option: send more
unassigned	>2	Unassigned

4.20.5 ZeroMQ Socket Type

Definition of the ZeroMQ Socket Type enumeration is dependent on definition of the ZMQ_CPP11, ZMQ_BUILD_DRAFT_API, ZMQ_VERSION, and ZMQ_VERSION_MAJOR symbolic constants.

Table 4-87: ZeroMQ Socket Type (ZMQ_CPP11 Defined) defines ZeroMQ Socket Type enumerated integer values when ZMQ_CPP11 is defined.

Table 4-87: ZeroMQ Socket Type (ZMQ_CPP11 Defined)

Mnemonic	Value	Description
unassigned	<0	Unassigned
Pair	ZMQ_PAIR (0)	Socket type: pair
pub	ZMQ_PUB (1)	Socket type: publish
sub	ZMQ_SUB (2)	Socket type: subscribe
req	ZMQ_REQ (3)	Socket type: request
rep	ZMQ_REP (4)	Socket type: reply
dealer	ZMQ_DEALER (5)	Socket type: dealer
router	ZMQ_ROUTER (6)	Socket type: router
pull	ZMQ_PULL (7)	Socket type: pull
push	ZMQ_PUSH (8)	Socket type: push
xpub	ZMQ_XPUB (9)	Socket type: raw publish
xsub	ZMQ_XSUB (10)	Socket type: raw subscribe

Table 4-88: ZeroMQ Receive Flags (ZMQ_CPP11 Defined AND ZMQ_BUILD_DRAFT_API Defined AND ZMQ_VERSION ≥ ZMQ_MAKE_VERSION(4, 2, 0)) defines ZeroMQ Socket Type enumerated integer values when the following are TRUE:

- ZMQ_CPP11 is defined
- ZMQ_BUILD_DRAFT_API is defined
- ZMQ_VERSION ≥ ZMQ_MAKE_VERSION(4, 2, 0)

Table 4-88: ZeroMQ Receive Flags (ZMQ_CPP11 Defined AND ZMQ_BUILD_DRAFT_API Defined AND ZMQ_VERSION ≥ ZMQ_MAKE_VERSION(4, 2, 0))

Mnemonic	Value	Description
server	ZMQ_SERVER (12)	Socket type: server
client	ZMQ_CLIENT (13)	Socket type: client
radio	ZMQ_RADIO (14)	Socket type: radio
dish	ZMQ_DISH (15)	Socket type: dish

Table 4-89: ZeroMQ Receive Flags (ZMQ_CPP11 Defined AND ZMQ_VERSION_MAJOR ≥ 4) defines ZeroMQ Socket Type enumerated integer values when the following are TRUE:

- ZMQ_CPP11 is defined
- ZMQ_VERSION_MAJOR ≥ 4

Table 4-89: ZeroMQ Receive Flags (ZMQ_CPP11 Defined AND ZMQ_VERSION_MAJOR ≥ 4)

Mnemonic	Value	Description
stream	ZMQ_STREAM (11)	Socket type: stream

5.0 Data Types

The following section defines data types used in HDTN software.

5.1 Bundle Protocol Receive File Data Types

The following sections define Bundle Protocol Receive (BPReceive) File data types.

5.1.1 BPReceive File Filename to Write Information Map

Table 5-1: BPReceive File Filename to Write Information Map Type defines the BPReceive File Filename to Write Information Map data type.

Table 5-1: BPReceive File Filename to Write Information Map Type

Mnemonic	Description	Type	Size (bytes)
filename_to_writeinfo_map_t	BPReceive filename to write information map	map	variable
filename_to_writeinfo_map_t.Key	BPReceive filename to write information map key	Boost Filesystem Path	variable
filename_to_writeinfo_map_t.T	BPReceive filename to write information map data type	BP Receive File Fragments Output File Stream Pair	N/A

5.1.2 BPReceive File Fragments Output File Stream Pair

Table 5-2: BPReceive File Fragments Output File Stream Pair Type defines the BPReceive File Fragments Output File Stream Pair data type.

Table 5-2: BPReceive File Fragments Output File Stream Pair Type

Mnemonic	Description	Type	Size (bytes)
fragments_ofstream_pair_t	BPReceive fragments output file stream pair	pair	variable
fragments_ofstream_pair_t.first	BPReceive fragments output file stream pair first element	Data Fragment Set	variable
fragments_ofstream_pair_t.second	BPReceive fragments output file stream pair second element	Boost Filesystem Output File Stream pointer	N/A

5.2 Bundle Protocol Sink Pattern Data Types

The following sections define Bundle Protocol Sink (BPSink) Pattern data types.

5.2.1 BPSink Pattern Bundle ID Final Destination Endpoint Identifier Pair

Table 5-3: BPSink Pattern Bundle ID Final Destination EID Pair Type defines the BPSink Pattern Bundle ID Final Destination Endpoint Identifier (EID) Pair data type.

Table 5-3: BPSink Pattern Bundle ID Final Destination EID Pair Type

Mnemonic	Description	Type	Size (bytes)
bundleid_finaldesteid_pair_t	BPSink pattern bundle ID final destination EID pair	pair	24
bundleid_finaldesteid_pair_t.first	BPSink pattern bundle ID final destination EID pair first element	uint64	8
bundleid_finaldesteid_pair_t.second	BPSink pattern bundle ID final destination EID pair second element	CBHE EID	16

5.2.2 BPSink Pattern Bundle User Data Pair

Table 5-4: BPSink Pattern Bundle User Data Pair Type defines the BPSink Pattern Bundle User Data Pair data type.

Table 5-4: BPSink Pattern Bundle User Data Pair Type

Mnemonic	Description	Type	Size (bytes)
bundle_userdata_pair_t	BPSink pattern bundle user data pair	pair	variable
bundle_userdata_pair_t.first	BPSink pattern bundle user data pair first element	uint8 vector	variable
bundle_userdata_pair_t.second	BPSink pattern bundle user data pair second element	BPSink Pattern Bundle ID Final Destination EID Pair	16

5.2.3 BPSink Pattern Destination EID Bundle Pair

Table 5-5: BPSink Pattern Destination EID Bundle Pair Type defines the BPSink Pattern Destination EID Bundle Pair data type.

Table 5-5: BPSink Pattern Destination EID Bundle Pair Type

Mnemonic	Description	Type	Size (bytes)
desteid_bundle_pair_t	BPSink pattern destination EID bundle pair	pair	variable
desteid_bundle_pair_t.first	BPSink pattern destination EID bundle pair first element	CBHE EID	16
desteid_bundle_pair_t.second	BPSink pattern destination EID bundle pair second element	uint8 vector	variable

5.3 Bundle Protocol Source Pattern Data Types

The following sections define BPSource Pattern data types.

5.3.1 BPSource Pattern Bundle ID Payload Size Pair

Table 5-6: BPSource Pattern Bundle ID Payload Size Pair Type defines the BPSource Pattern Bundle ID Payload Size Pair data type.

Table 5-6: BPSource Pattern Bundle ID Payload Size Pair Type

Mnemonic	Description	Type	Size (bytes)
bundleid_payloadsize_pair_t	BPSource pattern bundle ID payload size pair	pair	16
bundleid_payloadsize_pair_t.first	BPSource pattern bundle ID payload size pair first element	uint64	8
bundleid_payloadsize_pair_t.second	BPSource pattern bundle ID payload size pair second element	uint64	8

5.3.2 BPSource Pattern Bundle User Data Pair

Table 5-7: BPSource Pattern Bundle User Data Pair Type defines the BPSource Pattern Bundle User Data Pair data type.

Table 5-7: BPSource Pattern Bundle User Data Pair Type

Mnemonic	Description	Type	Size (bytes)
bundle_userdata_pair_t	Bundle user data pair	pair	variable
bundle_userdata_pair_t.first	Bundle user data pair first element	uint8 vector	variable
bundle_userdata_pair_t.second	Bundle user data pair second element	BPSource Pattern Bundle ID Payload Size Pair	16

5.3.3 BPSource Pattern Compressed Bundle Header Encoding Bundle Universally Unique Identifier Set

Table 5-8: BPSource Pattern CBHE Bundle UUID Set Type defines the BPSource Pattern Compressed Bundle Header Encoding (CBHE) Bundle Universally Unique Identifier (UUID) Set data type.

Table 5-8: BPSource Pattern CBHE Bundle UUID Set Type

Mnemonic	Description	Type	Size (bytes)
m_cbheBundleUuidSet	BPSource pattern CBHE bundle UUID set	set	variable
m_cbheBundleUuidSet.Key	BPSource pattern CBHE bundle UUID set key	CBHE Bundle UUID – No Fragment	32

5.3.4 BPSource Pattern Currently Sending Bundle ID Unordered Set

Table 5-9: BPSource Pattern Currently Sending Bundle ID Unordered Set Type defines the BPSource Pattern Currently Sending Bundle ID Unordered Set data type.

Table 5-9: BPSource Pattern Currently Sending Bundle ID Unordered Set Type

Mnemonic	Description	Type	Size (bytes)
m_currentlySendingBundleIdSet	Currently sending bundle ID unordered set	unordered set	variable
m_currentlySendingBundleIdSet.Key	Currently sending bundle ID unordered set key	uint64	8

5.4 Bundle Protocol Version 6 Data Types

The following sections define BPv6 data types.

5.4.1 BPv6 ACS Array

Table 5-10: BPv6 ACS Array Type defines the BPv6 ACS Array data type.

Table 5-10: BPv6 ACS Array Type

Mnemonic	Description	Type	Size (bytes)
acs_array_t	BPv6 ACS array	array	variable
acs_array_t.T	BPv6 ACS array class	BPv6 Administrative Record Content ACS	N/A
acs_array_t.N	BPv6 ACS array number of elements	NUM_ACS_STATUS_INDICES	N/A

5.4.2 BPv6 Canonical Block View List

Table 5-11: BPv6 Bundle View Canonical Block View List Type defines the BPv6 Canonical Block View List data type.

Table 5-11: BPv6 Bundle View Canonical Block View List Type

Mnemonic	Description	Type	Size (bytes)
m_listCanonicalBlockView	BPv6 canonical block view list	list	variable
m_listCanonicalBlockView.T	BPv6 canonical block view list data type	BPv6 Canonical Block View	N/A

5.4.3 BPv6 Custody Transfer Manager Custodian to ACS Array Map

Table 5-12: BPv6 Custody Transfer Manager Custodian to ACS Array Map Type defines the BPv6 Custody Transfer Manager Custodian to ACS Array Map data type.

Table 5-12: BPv6 Custody Transfer Manager Custodian to ACS Array Map Type

Mnemonic	Description	Type	Size (bytes)
m_mapCustodianToAcsArray	BPv6 custodian to ACS array map	map	variable
m_mapCustodianToAcsArray.Key	BPv6 custodian to ACS array map key	CBHE EID	16
m_mapCustodianToAcsArray.T	BPv6 custodian to ACS array map data type	ACS Array	variable

5.5 Bundle Protocol Version 7 Data Types

The following sections define BPv7 data types.

5.5.1 BPv7 Bundle Status Information Array

Table 5-13: BPv7 Status Information Array Type defines the BPv7 Bundle Status Information Array data type.

Table 5-13: BPv7 Status Information Array Type

Mnemonic	Description	Type	Size (bytes)
bundle_status_information_t	BPv7 bundle status information array	array	36
bundle_status_information_t.T	BPv7 bundle status information array class	BPv7 Status Information Content Pair	9
bundle_status_information_t.N	BPv7 bundle status information array number of elements (4)	size	N/A

5.5.2 BPv7 Canonical Block View List

Table 5-14: BPv7 Bundle View Canonical Block View List Type defines the BPv7 Canonical Block View List data type.

Table 5-14: BPv7 Bundle View Canonical Block View List Type

Mnemonic	Description	Type	Size (bytes)
m_listCanonicalBlockView	BPv7 canonical block view list	list	variable
m_listCanonicalBlockView.T	BPv7 canonical block view list data type	BPv7 Canonical Block View	N/A

5.5.3 BPv7 Encrypted Block Number To BCB Pointer Map

Table 5-15: BPv7 Encrypted Block Number To BCB Pointer Map Type defines the BPv7 Encrypted Block Number To BCB Pointer Map data type.

Table 5-15: BPv7 Encrypted Block Number To BCB Pointer Map Type

Mnemonic	Description	Type	Size (bytes)
m_mapEncryptedBlockNumberToBcbPtr	BPv7 encrypted block number to BDB pointer map	map	variable
m_mapEncryptedBlockNumberToBcbPtr.Key	BPv7 encrypted block number to BDB pointer map key	uint64	8
m_mapEncryptedBlockNumberToBcbPtr.T	BPv7 encrypted block number to BDB pointer map data type	BPv7 Block Confidentiality Block pointer	N/A

5.5.4 BPv7 ID Type

Table 5-16: BPv7 ID Type defines the BPv7 ID data type.

Table 5-16: BPv7 ID Type

Mnemonic	Description	Type	Size (bytes)
id_t	BPv7 ID	uint64	8

5.5.5 BPv7 ID Value Pair Type

Table 5-17: BPv7 ID Value Pair Type defines the BPv7 ID Value Pair data type.

Table 5-17: BPv7 ID Value Pair Type

Mnemonic	Description	Type	Size (bytes)
id_value_pair_t	BPv7 ID value pair	pair	variable
id_value_pair_t.first	BPv7 ID value pair first element	BPv7 ID	8
id_value_pair_t.second	BPv7 ID value pair second element	BPv7 Value Pointer	N/A

5.5.6 BPv7 ID Value Pairs Vector Type

Table 5-18: BPv7 ID Value Pairs Vector Type defines the BPv7 ID Value Pairs Vector data type.

Table 5-18: BPv7 ID Value Pairs Vector Type

Mnemonic	Description	Type	Size (bytes)
id_value_pairs_vec_t	BPv7 ID value pairs vector	BPv7 ID Value Pair vector	variable

5.5.7 BPv7 Parameter ID Type

Table 5-19: BPv7 Parameter ID Type defines the BPv7 Parameter ID data type.

Table 5-19: BPv7 Parameter ID Type

Mnemonic	Description	Type	Size (bytes)
parameter_id_t	BPv7 parameter ID	BPv7 ID	8

5.5.8 BPv7 Parameter Value Type

Table 5-20: BPv7 Parameter Value Type defines the BPv7 Parameter Value data type.

Table 5-20: BPv7 Parameter Value Type

Mnemonic	Description	Type	Size (bytes)
parameter_value_t	BPv7 parameter value	BPv7 Value Pointer	N/A

5.5.9 BPv7 Security Context Flags Type

Table 5-21: BPv7 Security Context Flags Type defines the BPv7 Security Context Flags data type.

Table 5-21: BPv7 Security Context Flags Type

Mnemonic	Description	Type	Size (bytes)
security_context_flags_t	BPv7 security context flags	uint8	1

5.5.10 BPv7 Security Context ID Type

Table 5-22: BPv7 Security Context ID Type defines the BPv7 Security Context ID data type.

Table 5-22: BPv7 Security Context ID Type

Mnemonic	Description	Type	Size (bytes)
security_context_id_t	BPv7 security context ID	uint64	8

5.5.11 BPv7 Security Context Parameter Type

Table 5-23: BPv7 Security Context Parameter Type defines the BPv7 Security Context Parameter data type.

Table 5-23: BPv7 Security Context Parameter Type

Mnemonic	Description	Type	Size (bytes)
security_context_parameter_t	BPv7 security context parameter	BPv7 ID Value Pair	variable

5.5.12 BPv7 Security Context Parameters Type

Table 5-24: BPv7 Security Context Parameters Type defines the BPv7 Security Context Parameters data type.

Table 5-24: BPv7 Security Context Parameters Type

Mnemonic	Description	Type	Size (bytes)
security_context_parameters_t	BPv7 security context parameters	BPv7 ID Value Pairs Vector	variable

5.5.13 BPv7 Security Result ID Type

Table 5-25: BPv7 Security Result ID Type defines the BPv7 Security Result ID data type.

Table 5-25: BPv7 Security Result ID Type

Mnemonic	Description	Type	Size (bytes)
security_result_id_t	BPv7 security result ID	BPv7 ID	8

5.5.14 BPv7 Security Result Type

Table 5-26: BPv7 Security Result Type defines the BPv7 Security Result data type.

Table 5-26: BPv7 Security Result Type

Mnemonic	Description	Type	Size (bytes)
security_result_t	BPv7 security result	BPv7 ID Value Pair	variable

5.5.15 BPv7 Security Result Value Type

Table 5-27: BPv7 Security Result Value Type defines the BPv7 Security Result Value data type.

Table 5-27: BPv7 Security Result Value Type

Mnemonic	Description	Type	Size (bytes)
security_result_value_t	BPv7 security result value	BPv7 Value Pointer	N/A

5.5.16 BPv7 Security Results Type

Table 5-28: BPv7 Security Results Type defines the BPv7 Security Results data type.

Table 5-28: BPv7 Security Results Type

Mnemonic	Description	Type	Size (bytes)
security_results_t	BPv7 security results	BPv7 ID Value Pairs Vector	variable

5.5.17 BPv7 Security Targets Type

Table 5-29: BPv7 Security Targets Type defines the BPv7 Security Targets data type.

Table 5-29: BPv7 Security Targets Type

Mnemonic	Description	Type	Size (bytes)
security_targets_t	BPv7 security targets	uint64 vector	variable

5.5.18 BPv7 Value Pointer Type

Table 5-30: BPv7 Value Pointer Type defines the BPv7 Value Pointer data type.

Table 5-30: BPv7 Value Pointer Type

Mnemonic	Description	Type	Size (bytes)
value_ptr_t	BPv7 value pointer	BPv7 Abstract Security Block Value Base unique pointer	N/A

5.5.19 BPv7 Status Information Content Pair

Table 5-31: BPv7 Status Information Content Pair Type defines the BPv7 Status Information Content Pair data type.

Table 5-31: BPv7 Status Information Content Pair Type

Mnemonic	Description	Type	Size (bytes)
status_info_content_t	BPv7 status information content pair	pair	9
status_info_content_t.first	BPv7 status information content pair first element	bool	1
status_info_content_t.second	BPv7 status information content pair second element	uint64	8

5.6 Bundle Protocol Security Policy Manager Data Types

The following sections define BPSec Policy Manager data types.

5.6.1 BPSec Policies By Role Array Type

Table 5-32: BPSec Policies By Role Array defines the BPSec Policies By Role Array data type.

Table 5-32: BPSec Policies By Role Array

Mnemonic	Description	Type	Size (bytes)
BpSecPoliciesByRoleArray	BPSec policies by role array	array	variable
BpSecPoliciesByRoleArray.T	BPSec policies by role array class	BPSec Policy shared pointer	N/A
BpSecPoliciesByRoleArray.N	BPSec policies by role array number of elements	RESERVED_MAX_ROLE_TYPES	N/A

5.6.2 BPSec Policy Shared Pointer Type

Table 5-33: BPSec Policy Shared Pointer defines the BPSec Policy Shared Pointer data type.

Table 5-33: BPSec Policy Shared Pointer

Mnemonic	Description	Type	Size (bytes)
BpSecPolicySharedPtr	BPSec policy shared pointer	BPSec Policy shared pointer	N/A

5.6.3 EID to Next Filter Map

Table 5-34: EID to Next Filter Map defines the EID to Next Filter Map data type.

Table 5-34: EID to Next Filter Map

Mnemonic	Description	Type	Size (bytes)
map_eid_to_next_filter_t	EID to next filter map	map	variable
map_eid_to_next_filter_t.Key	EID to next filter map key	CBHE EID	16
map_eid_to_next_filter_t.T	EID to next filter map data type	BPSec Policy Filter	N/A

5.6.4 Node ID to Next Filter Map

Table 5-35: Node ID to Next Filter Map defines the Node ID to Next Filter Map data type.

Table 5-35: Node ID to Next Filter Map

Mnemonic	Description	Type	Size (bytes)
map_node_id_to_next_filter_t	Node ID to next filter map	map	variable
map_node_id_to_next_filter_t.Key	Node ID to next filter map key	uint64	8
map_node_id_to_next_filter_t.T	Node ID to next filter map data type	BPSec Policy Filter	N/A

5.7 Bundle Storage Data Types

The following sections define Bundle Storage data types.

5.7.1 Segment ID Type

Definition of the Segment ID data type is dependent definition of the STORAGE_SEGMENT_ID_SIZE_BITS symbolic constant.

Table 5-36: Segment ID Type (STORAGE_SEGMENT_ID_SIZE_BITS = 32) defines the Segment ID data type when STORAGE_SEGMENT_ID_SIZE_BITS is 32.

Table 5-36: Segment ID Type (STORAGE_SEGMENT_ID_SIZE_BITS = 32)

Mnemonic	Description	Type	Size (bytes)
segment_id_t	Segment ID	uint32	4

Table 5-37: Segment ID Type (STORAGE_SEGMENT_ID_SIZE_BITS = 64) defines the Segment ID data type when STORAGE_SEGMENT_ID_SIZE_BITS is 64.

Table 5-37: Segment ID Type (STORAGE_SEGMENT_ID_SIZE_BITS = 64)

Mnemonic	Description	Type	Size (bytes)
segment_id_t	Segment ID	uint64	8

5.8 Bundle Storage Memory Manager Tree Array Data Types

The following sections define Bundle Storage Memory Manager Tree Array data types.

5.8.1 Memory Manager Tree Array Type

Table 5-38: Memory Manager Tree Array Type defines the Memory Manager Tree Array type.

Table 5-38: Memory Manager Tree Array Type

Mnemonic	Description	Type	Size (bytes)
segment_id_chain_vec_t	Segment ID chain vector	Segment ID vector	variable

5.8.2 Memory Manager Type

Table 5-39: Memory Manager Type defines the Memory Manager type.

Table 5-39: Memory Manager Type

Mnemonic	Description	Type	Size (bytes)
memmanager_t	Memory manager	uint64 vector vector	variable

5.9 Configuration Data Types

The following sections define Configuration data types.

5.9.1 BPSec Configuration Pointer

Table 5-40: BPSec Configuration Pointer defines the BPSec Configuration Pointer type.

Table 5-40: BPSec Configuration Pointer

Mnemonic	Description	Type	Size (bytes)
BpSecConfig_ptr	BPSec configuration pointer	BPSec Configuration shared pointer	N/A

5.9.2 Event Type to Event Set Pointer Lookup Table Type

Table 5-41: Event Type to Event Set Pointer Lookup Table Type defines the Event Type to Event Set Pointer Lookup Table type.

Table 5-41: Event Type to Event Set Pointer Lookup Table Type

Mnemonic	Description	Type	Size (bytes)
event_type_to_event_set_ptr_lut_t	Event type to event set pointer lookup table	array	variable
event_type_to_event_set_ptr_lut_t.T	Event type to event set pointer lookup table class	Security Operation Event Plus Actions Pair	3
event_type_to_event_set_ptr_lut_t.N	Event type to event set pointer lookup table number of elements	RESERVED_MAX_EVENTS	N/A

5.9.3 Policy Rules Vector

Table 5-42: Policy Rules Vector defines the Policy Rules Vector type.

Table 5-42: Policy Rules Vector

Mnemonic	Description	Type	Size (bytes)
policy_rules_vector_t	Policy rules vector	Policy Rules vector	variable

5.9.4 Security Context Parameters Vector

Table 5-43: Security Context Parameters Vector defines the Security Context Parameters Vector type.

Table 5-43: Security Context Parameters Vector

Mnemonic	Description	Type	Size (bytes)
security_context_params_vector_t	Security context parameters vector	Security Context Parameter vector	variable

5.9.5 Security Failure Event Sets Set

Table 5-44: Security Failure Event Sets Set defines the Security Failure Event Sets Set type.

Table 5-44: Security Failure Event Sets Set

Mnemonic	Description	Type	Size (bytes)
security_failure_event_sets_set_t	Security failure event sets set	set	variable
security_failure_event_sets_set_t.Key	Security failure event sets set key	Security Failure Event Sets	variable

5.9.6 Security Operation Event Plus Actions Pairs Vector

Table 5-45: Security Operation Event Plus Actions Pairs Vector Type defines the Security Operation Event Plus Actions Pairs Vector type.

Table 5-45: Security Operation Event Plus Actions Pairs Vector Type

Mnemonic	Description	Type	Size (bytes)
security_operation_event_plus_actions_pairs_vec_t	Security operation event plus actions pairs vector	Security Operation Event Plus Actions Pair vector	variable

5.10 Contact Graph Routing Data Types

The following sections define CGR data types.

5.10.1 CGR Contact Multigraph Routing Node Unordered Map

Table 5-46: CGR CMR Node Unordered Map Type defines the CGR Contact Multigraph Routing (CMR) Node Unordered Map data type.

Table 5-46: CGR CMR Node Unordered Map Type

Mnemonic	Description	Type	Size (bytes)
cmr_node_map_t	CGR CMR node unordered map	unordered map	variable
cmr_node_map_t.Key	CGR CMR node unordered map key	CGR Node ID	8
cmr_node_map_t.T	CGR CMR node unordered map data type	CGR CMR Map Data	N/A

5.10.2 CGR Node ID Type

Table 5-47: CGR Node ID Type defines the CGR Node ID data type.

Table 5-47: CGR Node ID Type

Mnemonic	Description	Type	Size (bytes)
nodeID_t	Node ID	uint64	8

5.10.3 CGR Route Visited Map

Table 5-48: CGR Route Visited Map Type defines the CGR Route Visited Map data type.

Table 5-48: CGR Route Visited Map Type

Mnemonic	Description	Type	Size (bytes)
__visited	CGR route visited map	map	variable
__visited.Key	CGR route visited map key	CGR Node ID	8
__visited.T	CGR route visited map data type	bool	1

5.10.4 CGR Vertex Adjacencies Unordered Map

Table 5-49: CGR Vertex Adjacencies Unordered Map Type defines the CGR Vertex Adjacencies Unordered Map data type.

Table 5-49: CGR Vertex Adjacencies Unordered Map Type

Mnemonic	Description	Type	Size (bytes)
adjacencies	CGR vertex adjacencies unordered map	unordered map	variable
adjacencies.Key	CGR vertex adjacencies unordered map key	CGR Node ID	8
adjacencies.T	CGR vertex adjacencies unordered map data type	CGR Node ID vector	variable

5.10.5 CGR Vertex Pointer Plus Arrival Time Pair

Table 5-50: CGR Vertex Pointer Plus Arrival Time Pair Type defines the CGR Vertex Pointer Plus Arrival Time Pair data type.

Table 5-50: CGR Vertex Pointer Plus Arrival Time Pair Type

Mnemonic	Description	Type	Size (bytes)
vertex_ptr_plus_arrival_time_pair_t	CGR vertex pointer plus arrival time pair	pair	N/A
vertex_ptr_plus_arrival_time_pair_t.first	CGR vertex pointer plus arrival time pair first element	CGR Vertex pointer	N/A
vertex_ptr_plus_arrival_time_pair_t.second	CGR vertex pointer plus arrival time pair second element	time	variable

5.11 Custody ID Allocator Data Types

The following sections define Custody ID Allocator data types.

5.11.1 Custody ID Allocator Bundle Source EID to Next Custody Transfer Enhancement Block Custody ID Map

Table 5-51: Custody ID Allocator Bundle Source EID to Next CTEB Custody ID Map Type defines the Custody ID Allocator Bundle Source EID to Next Custody Transfer Enhancement Block (CTEB) Custody ID Map data type.

Table 5-51: Custody ID Allocator Bundle Source EID to Next CTEB Custody ID Map Type

Mnemonic	Description	Type	Size (bytes)
m_mapBundleSrcEidToNextCtebCustodyId	Custody ID allocator bundle source EID to next CTEB custody ID map	map	24
m_mapBundleSrcEidToNextCtebCustodyId.Key	Custody ID allocator bundle source EID to next CTEB custody ID map key	CBHE EID	16
m_mapBundleSrcEidToNextCtebCustodyId.T	Custody ID allocator bundle source EID to next CTEB custody ID map data type	uint64	8

5.12 HDTN Configuration Data Types

The following sections define HDTN Configuration data types.

5.12.1 HDTN Configuration Pointer Type

Table 5-52: HDTN Configuration Pointer Type defines the HDTN Configuration Pointer data type.

Table 5-52: HDTN Configuration Pointer Type

Mnemonic	Description	Type	Size (bytes)
HdtnConfig_ptr	HDTN configuration pointer	HDTN Configuration shared pointer	N/A

5.13 HDTN Distributed Configuration Data Types

The following sections define HDTN Distributed Configuration data types.

5.13.1 HDTN Distributed Configuration Pointer Type

Table 5-53: HDTN Distributed Configuration Pointer Type defines the HDTN Distributed Configuration Pointer data type.

Table 5-53: HDTN Distributed Configuration Pointer Type

Mnemonic	Description	Type	Size (bytes)
HdtnDistributedConfig_ptr	HDTN distributed configuration pointer	HDTN Distributed Configuration shared pointer	N/A

5.14 Induct Configuration Data Types

The following sections define Induct Configuration data types.

5.14.1 Induct Element Configuration Vector

Table 5-54: Induct Element Configuration Vector Type defines the Induct Element Configuration Vector data type.

Table 5-54: Induct Element Configuration Vector Type

Mnemonic	Description	Type	Size (bytes)
induct_element_config_vector	Inducts element configuration vector	Vector of Induct Element Configuration	variable

5.14.2 Inducts Configuration Pointer Type

Table 5-55: Inducts Configuration Pointer Type defines the Inducts Configuration pointer type.

Table 5-55: Inducts Configuration Pointer Type

Mnemonic	Description	Type	Size (bytes)
InductsConfig_ptr	Inducts configuration pointer	Inducts Configuration shared pointer	N/A

5.15 Induct Manager Data Types

The following sections define Induct Manager data types.

5.15.1 Induct List Type

Table 5-56: Induct List Type defines the Induct List data type.

Table 5-56: Induct List Type

Mnemonic	Description	Type	Size (bytes)
m_inductsList	Inducts list	list	variable
m_inductsList.T	Inducts list data type	Induct	N/A

5.16 Logger Data Types

The following sections define Logger data types.

5.16.1 Logger Sink Type

Table 5-57: Logger Sink Type defines the Logger Sink type.

Table 5-57: Logger Sink Type

Mnemonic	Description	Type	Size (bytes)
sink_t	Logger sink	Boost Log Sink Synchronous Sink	N/A

5.16.2 Process Attribute Type

Table 5-58: Process Attribute Type defines the Process Attribute type.

Table 5-58: Process Attribute Type

Mnemonic	Description	Type	Size (bytes)
process_attr_t	Process attribute	Boost Log Attributes Constant	N/A

5.16.3 Severity Channel Logger Type

Table 5-59: Severity Channel Logger Type defines the Logger Sink type.

Table 5-59: Severity Channel Logger Type

Mnemonic	Description	Type	Size (bytes)
severity_channel_logger_t	Severity channel logger	Boost Log Sources Severity Channel Logger Multithread	N/A

5.17 Common Data Types

The following section defines data types common across HDTN implementation.

5.17.1 Common Data Fragment No Overlap Allow Abut Set

Definition of the Common Data Fragment No Overlap Allow Abut Set data type is dependent definition of the FRAGMENT_SET_USE_FREE_LIST_ALLOCATOR symbolic constant.

Table 5-60: Common Data Fragment No Overlap Allow Abut Set Type (FRAGMENT_SET_USE_FREE_LIST_ALLOCATOR = TRUE) defines the Common Data Fragment No Overlap Allow Abut Set data type when FRAGMENT_SET_USE_FREE_LIST_ALLOCATOR is TRUE.

Table 5-60: Common Data Fragment No Overlap Allow Abut Set Type (FRAGMENT_SET_USE_FREE_LIST_ALLOCATOR = TRUE)

Mnemonic	Description	Type	Size (bytes)
data_fragment_no_overlap_allow_abut_set_t	Common data fragment no overlap allow abut set	set	variable
data_fragment_no_overlap_allow_abut_set_t.Key	Common data fragment no overlap allow abut set key	Common Data Fragment No Overlap Allow Abut	16
data_fragment_no_overlap_allow_abut_set_t.Compare	Common data fragment no overlap allow abut set compare	less	N/A
data_fragment_no_overlap_allow_abut_set_t.Allocator	Common data fragment no overlap allow abut set allocator	Common Free List Allocator	N/A

Table 5-61: Common Data Fragment No Overlap Allow Abut Set Type (Default) defines the default Common Data Fragment No Overlap Allow Abut Set data type.

Table 5-61: Common Data Fragment No Overlap Allow Abut Set Type (Default)

Mnemonic	Description	Type	Size (bytes)
data_fragment_no_overlap_allow_abut_set_t	Common data fragment no overlap allow abut set	set	variable
data_fragment_no_overlap_allow_abut_set_t.Key	Common data fragment no overlap allow abut set key	Common Data Fragment No Overlap Allow Abut	16

5.17.2 Common Data Fragment Set

Definition of the Common Data Fragment Set data type is dependent definition of the FRAGMENT_SET_USE_FREE_LIST_ALLOCATOR symbolic constant.

Table 5-62: Common Data Fragment Set Type (FRAGMENT_SET_USE_FREE_LIST_ALLOCATOR = TRUE) defines the Common Data Fragment Set data type when FRAGMENT_SET_USE_FREE_LIST_ALLOCATOR is TRUE.

Table 5-62: Common Data Fragment Set Type (FRAGMENT_SET_USE_FREE_LIST_ALLOCATOR = TRUE)

Mnemonic	Description	Type	Size (bytes)
data_fragment_set_t	Common data fragment set	set	variable
data_fragment_set_t.Key	Common data fragment set key	Common Data Fragment	16
data_fragment_set_t.Compare	Common data fragment set compare	less	N/A
data_fragment_set_t.Allocator	Common data fragment set allocator	Common Free List Allocator	N/A

Table 5-63: Common Data Fragment Set Type (Default) defines the default Common Data Fragment Set data type.

Table 5-63: Common Data Fragment Set Type (Default)

Mnemonic	Description	Type	Size (bytes)
data_fragment_set_t	Common data fragment set	set	variable
data_fragment_set_t.Key	Common data fragment set key	Common Data Fragment	16

5.17.3 Common Data Segment Pending Map

Definition of the Common Data Segment (DS) Pending Map data type is dependent definition of the FRAGMENT_SET_USE_FREE_LIST_ALLOCATOR symbolic constant.

Table 5-64: Common DS Pending Map Type (FRAGMENT_SET_USE_FREE_LIST_ALLOCATOR = TRUE) defines the Common DS Pending Map data type when FRAGMENT_SET_USE_FREE_LIST_ALLOCATOR is TRUE.

Table 5-64: Common DS Pending Map Type (FRAGMENT_SET_USE_FREE_LIST_ALLOCATOR = TRUE)

Mnemonic	Description	Type	Size (bytes)
ds_pending_map_t	DS pending map	map	variable
ds_pending_map_t.Key	DS pending map key	Common Data Fragment Unique Overlapping	N/A
ds_pending_map_t.T	DS pending map data type	uint64	8
ds_pending_map_t.Compare	DS pending map compare	less	N/A
ds_pending_map_t.Allocator	DS pending map allocator	Common Free List Allocator	N/A

Table 5-65: Common DS Pending Map Type (Default) defines the default Common DS Pending Map data type.

Table 5-65: Common DS Pending Map Type (Default)

Mnemonic	Description	Type	Size (bytes)
ds_pending_map_t	DS pending map	map	variable
ds_pending_map_t.Key	DS pending map key	Common Data Fragment Unique Overlapping	N/A

5.17.4 Common LTP Checkpoint Serial Number Active Timers List

Table 5-66: Common LTP CSN Active Timers List Type defines the Common LTP Checkpoint Serial Number (CSN) Active Timers List data type.

Table 5-66: Common LTP CSN Active Timers List Type

Mnemonic	Description	Type	Size (bytes)
checkpoint_serial_number_active_timers_list_t	Checkpoint serial number active timers list	list	variable
checkpoint_serial_number_active_timers_list_t.T	Checkpoint serial number active timers list data type	uint64	8
checkpoint_serial_number_active_timers_list_t.Allocator	Checkpoint serial number active timers list allocator	Common Free List Allocator	N/A

5.17.5 Common LTP Checkpoint Serial Number Is Secondary Pair

Table 5-67: Common LTP CSN Is Secondary Pair Type defines the Common LTP CSN Is Secondary Pair data type.

Table 5-67: Common LTP CSN Is Secondary Pair Type

Mnemonic	Description	Type	Size (bytes)
csn_issecondary_pair_t	CSN is secondary pair	pair	9
csn_issecondary_pair_t.first	CSN is secondary pair first element	uint64	8
csn_issecondary_pair_t.second	CSN is secondary pair second element	bool	1

5.17.6 Common LTP Checkpoint Serial Numbers Received Set

Table 5-68: Common LTP CSN Received Set Type defines the Common LTP CSN Received Set data type.

Table 5-68: Common LTP CSN Received Set Type

Mnemonic	Description	Type	Size (bytes)
checkpoint_serial_numbers_received_set_t	Checkpoint serial numbers received set	set	variable
checkpoint_serial_numbers_received_set_t.Key	Checkpoint serial numbers received set key	uint64	8
checkpoint_serial_numbers_received_set_t.Compare	Checkpoint serial numbers received set compare	less	N/A
checkpoint_serial_numbers_received_set_t.Allocator	Checkpoint serial numbers received set allocator	Common Free List Allocator	N/A

5.17.7 Common LTP Report Segment Pending Map

Table 5-69: Common LTP RS Pending Map Type defines the Common LTP Report Segment (RS) Pending Map data type.

Table 5-69: Common LTP RS Pending Map Type

Mnemonic	Description	Type	Size (bytes)
rs_pending_map_t	RS pending map	map	variable
rs_pending_map_t.Key	RS pending map key	Common Data Fragment No Overlap Allow Abut	N/A
rs_pending_map_t.T	RS pending map data type	Common LTP CSN Is Secondary Pair	N/A
rs_pending_map_t.Compare	RS pending map compare	less	N/A
rs_pending_map_t.Allocator	RS pending map allocator	Common Free List Allocator	N/A

5.17.8 Common LTP Report Segment Serial Numbers Received Set

Table 5-70: Common LTP RS Serial Numbers Received Set Type defines the Common LTP RS Serial Numbers Received Set data type.

Table 5-70: Common LTP RS Serial Numbers Received Set Type

Mnemonic	Description	Type	Size (bytes)
report_segment_serial_numbers_received_set_t	Report segment serial numbers received set	set	variable
report_segment_serial_numbers_received_set_t.Key	Report segment serial numbers received set key	uint64	8
report_segment_serial_numbers_received_set_t.Compare	Report segment serial numbers received set compare	less	N/A
report_segment_serial_numbers_received_set_t.Allocator	Report segment serial numbers received set allocator	Common Free List Allocator	N/A

5.17.9 Common LTP Report Segments Sent Map

Table 5-71: Common LTP RS Sent Map Type defines the Common LTP RS Sent Map data type.

Table 5-71: Common LTP RS Sent Map Type

Mnemonic	Description	Type	Size (bytes)
report_segments_sent_map_t	Report segments sent map	map	variable
report_segments_sent_map_t.Key	Report segments sent map key	uint64	8
report_segments_sent_map_t.T	Report segments sent map data type	LTP Report Segment	variable
report_segments_sent_map_t.Compare	Report segments sent map compare	less	N/A
report_segments_sent_map_t.Allocator	Report segments sent map allocator	Common Free List Allocator	N/A

5.17.10 Common LTP Report Serial Number Active Timers List Type

Table 5-72: Common LTP Report Serial Number Active Timers List Type defines the Common LTP Report Serial Number Active Timers List type.

Table 5-72: Common LTP Report Serial Number Active Timers List Type

Mnemonic	Description	Type	Size (bytes)
report_serial_number_active_timers_list_t	Report serial number active timers list	list	variable
report_serial_number_active_timers_list_t.T	Report serial number active timers list data type	uint64	8
report_serial_number_active_timers_list_t.Allocator	Report serial number active timers list allocator	Common Free List Allocator	N/A

5.17.11 Common LTP Retry Count Iterator Pair

Table 5-73: Common LTP Retry Count Iterator Pair Type defines the Common LTP Retry Count Iterator Pair data type.

Table 5-73: Common LTP Retry Count Iterator Pair Type

Mnemonic	Description	Type	Size (bytes)
it_retrycount_pair_t	Retry count iterator pair	pair	N/A
it_retrycount_pair_t.first	Retry count iterator pair first element	Common LTP Report Segments Sent Map Constant Iterator	N/A
it_retrycount_pair_t.second	Retry count iterator pair second element	uint32	4

5.17.12 Common LTP Session Receiver Recycled Data Unique Pointer Type

Table 5-74: Common LTP Session Receiver Recycled Data Unique Pointer Type defines the Common LTP Session Receiver Recycled Data Unique Pointer type.

Table 5-74: Common LTP Session Receiver Recycled Data Unique Pointer Type

Mnemonic	Description	Type	Size (bytes)
LtpSessionReceiverRecycledDataUniquePtr	LTP session receiver recycled data unique pointer	Common LTP Session Receiver Recycled Data unique pointer	N/A

5.17.13 Common LTP Session Receiver Recycler Type

Table 5-75: Common LTP Session Receiver Recycler Type defines the Common LTP Session Receiver Recycler type.

Table 5-75: Common LTP Session Receiver Recycler Type

Mnemonic	Description	Type	Size (bytes)
LtpSessionReceiverRecycler	LTP session receiver recycler	Common LTP Session Receiver Recycled Data Unique Pointer User Data Recycler	N/A

5.17.14 Common LTP Session Sender Recycled Data Unique Pointer Type

Table 5-76: Common LTP Session Sender Recycled Data Unique Pointer Type defines the Common LTP Session Sender Recycled Data Unique Pointer type.

Table 5-76: Common LTP Session Sender Recycled Data Unique Pointer Type

Mnemonic	Description	Type	Size (bytes)
LtpSessionSenderRecycledDataUniquePtr	LTP session sender recycled data unique pointer	Common LTP Session Sender Recycled Data unique pointer	N/A

5.17.15 Common LTP Session Sender Recycler Type

Table 5-77: Common LTP Session Sender Recycler Type defines the Common LTP Session Sender Recycler type.

Table 5-77: Common LTP Session Sender Recycler Type

Mnemonic	Description	Type	Size (bytes)
LtpSessionSenderRecycler	LTP session sender recycler	Common LTP Session Sender Recycled Data Unique Pointer User Data Recycler	N/A

5.17.16 Common Telemetry Bundle Count Plus Bundle Bytes Pair

Table 5-78: Common Telemetry Bundle Count Plus Bundle Bytes Pair Type defines the Common Telemetry Bundle Count Plus Bundle Bytes Pair data type.

Table 5-78: Common Telemetry Bundle Count Plus Bundle Bytes Pair Type

Mnemonic	Description	Type	Size (bytes)
bundle_count_plus_bundle_bytes_pair_t	Bundle count plus bundle bytes pair	pair	16
bundle_count_plus_bundle_bytes_pair_t.first	Bundle count plus bundle bytes pair first element	uint64	8
bundle_count_plus_bundle_bytes_pair_t.second	Bundle count plus bundle bytes pair second element	uint64	8

5.17.17 Common Telemetry Node ID to Expiring Before Threshold Count Map

Table 5-79: Common Telemetry Node ID to Expiring Before Threshold Count Map Type defines the Common Telemetry Node ID to Expiring Before Threshold Count Map data type.

Table 5-79: Common Telemetry Node ID to Expiring Before Threshold Count Map Type

Mnemonic	Description	Type	Size (bytes)
mapNodeIdToExpiringBeforeThresholdCount	Node ID to expiring before threshold count map	map	variable
mapNodeIdToExpiringBeforeThresholdCount.Key	Node ID to expiring before threshold count map key	uint64	8
mapNodeIdToExpiringBeforeThresholdCount.T	Node ID to expiring before threshold count map data type	Common Telemetry Bundle Count Plus Bundle Bytes Pair	16

5.17.18 Common UINT8 Padded Vector

Table 5-80: Common UINT8 Padded Vector Type defines the Common UINT8 Padded Vector data type.

Table 5-80: Common UINT8 Padded Vector Type

Mnemonic	Description	Type	Size (bytes)
padded_vector_uint8_t	UINT8 padded vector	vector	variable
padded_vector_uint8_t.T	UINT8 padded vector data type	uint8	1
padded_vector_uint8_t.Allocator	UINT8 padded vector allocator	Padded Memory Allocator	N/A

6.0 Data Structures

The following section defines data structures used in HDTN software.

6.1 Basic Directory Monitor Data Structures

The following section defines data structures specific to Basic Directory Monitor implementation.

6.1.1 Directory Monitor Event

Table 6-1: Directory Monitor Event defines the Directory Monitor Event structure.

Table 6-1: Directory Monitor Event

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
path	Path to directory to be scanned/monitored	Boost Filesystem Path	variable	*	N/A	N/A	N/A
type	Type of files/directories to scan for	Directory Monitor Event Type	4	“null”*	“null”	“recursive _rescan”	N/A

6.2 Bidirectional Link Data Structures

The following section defines data structures specific to Bidirectional Link implementation.

6.2.1 Bidirectional Link Atomic Telemetry

Table 6-2: Bidirectional Link Atomic Telemetry defines the Bidirectional Link Atomic Telemetry structure.

Table 6-2: Bidirectional Link Atomic Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
totalFragmentsReceived	Total fragments received	atomic uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
totalBundlesReceived	Total bundles received	atomic uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
totalBundleBytesReceived	Total bundle bytes received	atomic uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
totalFragmentsSent	Total fragments sent	atomic uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
totalFragmentsSentAndAcked	Total fragments sent and acknowledged	atomic uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
totalBundlesSent	Total bundles sent	atomic uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
totalBundlesSentAndAcked	Total bundles sent and acknowledged	atomic uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
totalBundleBytesSent	Total bundle bytes sent	atomic uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
totalBundleBytesSentAndAcked	Total bundle bytes sent and acknowledged	atomic uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
totalBundlesFailedToSend	Total bundles failed to send	atomic uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
numTcpReconnectAttempts	Number of TCP reconnect attempts	atomic uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
linkIsUpPhysically	Link is up physically flag	bool	1	FALSE*	FALSE	TRUE	N/A

6.3 Bundle Protocol Generator Data Structures

This section defines data structures specific to Bundle Protocol Generator (BPGen) implementation.

6.3.1 BPGen Header

Table 6-3: BPGen Header defines the BPGen Header structure.

Table 6-3: BPGen Header

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
seq	Sequence	uint64	8	0	UINT64_MIN	UINT64_MAX	N/A
tsc	Time stamp counter	uint64	8	0	UINT64_MIN	UINT64_MAX	N/A
abstime	Absolute time	timespec	N/A		N/A	N/A	N/A

6.4 Bundle Protocol Ingest Data Structures

This section defines data structures specific to Bundle Protocol Ingest (BPIng) implementation.

6.4.1 BPIng Data

Definition of the BPIng Data structure is dependent on definition of the BOOST_DATE_TIME_POSIX_TIME_STD_CONFIG compiler option.

Table 6-4: BPIng Data (BOOST_DATE_TIME_POSIX_TIME_STD_CONFIG Not Defined) defines BPIng Data when BOOST_DATE_TIME_POSIX_TIME_STD_CONFIG is not defined.

Table 6-4: BPIng Data (BOOST_DATE_TIME_POSIX_TIME_STD_CONFIG Not Defined)

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
sequence	Sequence	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
sendTime	Send time	ptime	8	*	UINT64_MIN	UINT64_MAX	µsec

Table 6-5: BPIng Data (BOOST_DATE_TIME_POSIX_TIME_STD_CONFIG Defined) defines BPIng Data when BOOST_DATE_TIME_POSIX_TIME_STD_CONFIG is defined.

Table 6-5: BPIng Data (BOOST_DATE_TIME_POSIX_TIME_STD_CONFIG Defined)

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
sequence	Sequence	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
sendTime	Send time	ptime	12	*	0	$2^{96} - 1$	nsec

6.5 BPReceive File Data Structures

This section defines data structures specific to BPReceive implementation.

6.5.1 BPReceive File Send File Metadata

Table 6-6: BPReceive File Send File Metadata defines the BPReceive File Send File Metadata structure.

Table 6-6: BPReceive File Send File Metadata

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
totalFileSize	Total file size	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
fragmentOffset	Frame offset	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
fragmentLength	Fragment length	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
pathLen	Path length	uint8	1	0*	UINT8_MIN	UINT8_MAX	bytes
unused1	Unused	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
unused2	Unused	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
unused3	Unused	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A

6.6 Bundle Protocol Receive Stream Data Structures

This section defines data structures specific to BPReceive Stream implementation.

6.6.1 BPReceive Stream Parameters

Table 6-7: BPReceive Stream Parameters defines the BPReceive Stream Parameters structure.

Table 6-7: BPReceive Stream Parameters

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
rtpDestHostname	RTP destination host name	string	variable	“”*	N/A	N/A	N/A
rtpDestPort	RTP destination port	uint64	8	0*	UINT64_MIN	UINT64_ MAX	N/A
maxOutgoingRtp PacketSizeBytes	Max outgoing RTP packet size in bytes	uint64	8	0*	UINT64_MIN	UINT64_ MAX	bytes
outductType	Outduct type	uint64	8	0*	UINT64_MIN	UINT64_ MAX	N/A
shmSocketPath	SHM socket path	string	variable	“”*	N/A	N/A	N/A
gstCaps	GST caps	string	variable	“”*	N/A	N/A	N/A

6.7 Bundle Protocol Security Bundle Processor Data Structures

This section defines data structures specific to BPSec Bundle Processor implementation.

6.7.1 Envelope Cipher Context Wrapper

Table 6-8: Envelope Cipher Context Wrapper defines the Envelope Cipher Context Wrapper structure.

Table 6-8: Envelope Cipher Context Wrapper

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_ctx	Context	Envelope cipher context pointer	N/A	*	N/A	N/A	N/A

6.7.2 Hash Message Authentication Code Context Wrapper

Definition of the HMAC Context Wrapper structure is dependent on definition of the BPSEC_USE_SSL3 symbolic constant.

Table 6-9: HMAC Context Wrapper – BPSEC_USE_SSL3 Defined defines the HMAC Context Wrapper structure when BPSEC_USE_SSL3 is defined.

Table 6-9: HMAC Context Wrapper – BPSEC_USE_SSL3 Defined

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_mac	MAC Pointer	Envelope message authentication code pointer	N/A	*	N/A	N/A	N/A
m_ctx	Context Pointer	Envelope message authentication code context pointer	N/A	*	N/A	N/A	N/A

Table 6-10: HMAC Context Wrapper – Default defines the default HMAC Context Wrapper structure.

Table 6-10: HMAC Context Wrapper – Default

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_ctx	Context Pointer	Hash message authentication code context pointer	N/A	*	N/A	N/A	N/A

6.7.3 Reusable Elements Internal

Table 6-11: Reusable Elements Internal defines the Reusable Elements Internal structure.

Table 6-11: Reusable Elements Internal

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
constBufferVec	Constant Buffer Vector	Boost Asynchronous Input/Output (ASIO) Constant Buffer	variable	*	N/A	N/A	N/A
verifyOnlyDecryption TemporaryMemory	Verify Only Decryption Temporary Memory	uint8 vector	variable	*	N/A	N/A	N/A

6.8 Bundle Protocol Security Policy Manager Data Structures

This section defines data structures specific to BPSec Policy Manager implementation.

6.8.1 BPSec Policy

Table 6-12: BPSec Policy defines the BPSec Policy structure.

Table 6-12: BPSec Policy

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_doIntegrity	Do integrity flag	bool	1	FALSE*	FALSE	TRUE	N/A
m_doConfidentiality	Do confidentiality flag	bool	1	FALSE*	FALSE	TRUE	N/A
m_bcbTargetsPayloadBlock	BCB targets payload block flag	bool	1	FALSE*	FALSE	TRUE	N/A
m_bibMustBeEncrypted	BIB must be encrypted flag	bool	1	FALSE*	FALSE	TRUE	N/A
m_integrityVariant	Integrity variant	COSE Algorithms	4	*unassigned**	“A128GCM”	“HMAC_512_512”	N/A
m_integrityScopeMask	Integrity scope mask	BPSec BIB-HMAC-SHA2 Integrity Scope Flags	4	0x0000 0000*	N/A	N/A	N/A

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_bibCrcType	BIB CRC type	BPv7 CRC Type	1	“NONE”*	“NONE”	“CRC32C”	N/A
m_bibBlockTypeTargets	BIB block type targets	Common Data Fragment Set	variable	*	N/A	N/A	N/A
m_hmacKeyEncryptionKey	HMAC key encryption key	uint8 vector	variable	*	N/A	N/A	N/A
m_hmacKey	HMAC key	uint8 vector	variable	*	N/A	N/A	N/A
m_integritySecurityFailureEventSetReferencePtr	Integrity security failure event set reference pointer	Security Failure Event Sets pointer	N/A	*	N/A	N/A	N/A
m_confidentialityVariant	Confidentiality variant	COSE Algorithms	4	*unassigned**	“A128GCM”	“HMAC_512_512”	N/A
m_use12ByteIv	Use 12-Byte IV flag	bool	1	FALSE*	FALSE	TRUE	N/A
m_aadScopeMask	AAD scope mask	BPSec BCB-AES-GCM Additional Authentication Data Scope Flags	4	0x0000 0000*	N/A	N/A	N/A
m_bcbCrcType	BCB CRC type	BPv7 CRC Type	1	“NONE”*	“NONE”	“CRC32C”	N/A
m_bcbBlockTypeTargets	BCB block type targets	Common Data Fragment Set	variable	*	N/A	N/A	N/A
m_confidentialityKeyEncryptionKey	Confidentiality key encryption key	uint8 vector	variable	*	N/A	N/A	N/A
m_dataEncryptionKey	Data encryption key	uint8 vector	variable	*	N/A	N/A	N/A
m_confidentialitySecurityFailureEventSetReferencePtr	Confidentiality security failure event set reference pointer	Security Failure Event Sets pointer	N/A	*	N/A	N/A	N/A

6.8.2 BPSec Policy Filter

Table 6-13: BPSec Policy Filter defines the BPSec Policy Filter structure.

Table 6-13: BPSec Policy Filter

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_nodeIdToNextFilterMap	Node ID to next filter map	Node ID to Next Filter Map	variable	*	N/A	N/A	N/A
m_eidToNextFilterMap	EID to next filter map	EID to Next Filter Map	variable	*	N/A	N/A	N/A
m_anyEidToNextFilterPtr	Any EID to next filter pointer	BPSec Policy Filter pointer	N/A	*	N/A	N/A	N/A
m_policiesByRoleArray	Policies by role array	BPSec Policies By Role Array	variable	*	N/A	N/A	N/A

6.8.3 BPSec Policy Processing Context

Table 6-14: DBPSec Policy Processing Context defines the BPSec Policy Processing Context structure.

Table 6-14: DBPSec Policy Processing Context

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_ivStruct	Initialization vector structure	Initialization Vectors for One Thread	variable	*	N/A	N/A	N/A
m_bpsecReusableElementsInternal	BPSec reusable elements internal	Reusable Elements Internal	N/A	*	N/A	N/A	N/A
m_hmacCtxWrapper	HMAC context wrapper	Hash Message Authentication Code Context Wrapper	N/A	*	N/A	N/A	N/A
m_evpCtxWrapper	Envelope context wrapper	Envelope Cipher Context Wrapper	N/A	*	N/A	N/A	N/A
m_ctxWrapperKeyWrapOps	Context wrapper key wrap ops	Envelope Cipher Context Wrapper	N/A	*	N/A	N/A	N/A
m_bcbTargetBlockNumbers	BCB target block numbers	uint64 vector	variable	*	N/A	N/A	N/A
m_bibTargetBlockNumbers	BIB target block numbers	uint64 vector	variable	*	N/A	N/A	N/A
m_bcbTargetBibBlockNumberPlaceholderIndex	BCB target BIB block number placeholder index	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_tmpBlocks	Temporary blocks	BPv7 Canonical Block View vector	variable	*	N/A	N/A	N/A
m_searchCacheBcbAcceptor	Search cache BCB acceptor	Policy Search Cache	N/A	*	N/A	N/A	N/A
m_searchCacheBcbVerifier	Search cache BCB verifier	Policy Search Cache	N/A	*	N/A	N/A	N/A
m_searchCacheBibAcceptor	Search cache BIB acceptor	Policy Search Cache	N/A	*	N/A	N/A	N/A
m_searchCacheBibVerifier	Search cache BIB verifier	Policy Search Cache	N/A	*	N/A	N/A	N/A
m_searchCacheSource	Search cache source	Policy Search Cache	N/A	*	N/A	N/A	N/A

6.8.4 Policy Search Cache

Table 6-15: Policy Search Cache defines the Policy Search Cache structure.

Table 6-15: Policy Search Cache

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
securitySourceEid	Security source EID	CBHE EID	16	*	N/A	N/A	N/A
bundleSourceEid	Bundle source EID	CBHE EID	16	*	N/A	N/A	N/A
bundleFinalDestEid	Bundle final destination EID	CBHE EID	16	*	N/A	N/A	N/A
role	Role	BPSec Role	4	*	N/A	N/A	N/A
wasCacheHit	Was cache hit flag	bool	1	FALSE*	FALSE	TRUE	N/A
foundPolicy	Found policy	BPSec Policy pointer	N/A	*	N/A	N/A	N/A

6.9 Bundle Protocol Send Data Structures

This section defines data structures specific to BPSend implementation.

6.9.1 BPSend File Send File Metadata

Table 6-16: BPSend File Send File Metadata defines the BPSend File Send File Metadata structure.

Table 6-16: BPSend File Send File Metadata

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
totalFileSize	Total file size	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
fragmentOffset	Fragment offset	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
fragmentLength	Fragment length	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
pathLen	Path length	uint8	1	0*	UINT8_MIN	UINT8_MAX	bytes
unused1	Unused byte 1	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
unused2	Unused byte 2	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
unused3	Unused byte 3	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A

6.10 Bundle Protocol Sync Asynchronous Data Structures

This section defines data structures specific to Bundle Protocol Sink (BPSink) Asynchronous implementation.

6.10.1 BPSink Final Stats

Table 6-17: BPSink Final Stats defines the BPSink Final Stats structure.

Table 6-17: BPSink Final Stats

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_totalBytesRx	Total bytes received	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalBundlesRx	Total bundles received	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_receivedCount	Received count	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_duplicateCount	Duplicate count	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_seqHval	Sequence high value	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_seqBase	Sequence base	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.11 Bundle Protocol Version 6 Data Structures

This section defines data structures specific to BPv6 implementation.

6.11.1 BPv6 Administrative Record

Table 6-18: BPv6 Administrative Record defines the BPv6 Administrative Record structure.

Table 6-18: BPv6 Administrative Record

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
Bpv6CanonicalBlock	BPv6 canonical block	BPv6 Canonical Block	N/A	N/A	N/A	N/A	N/A
m_adminRecordTypeCode	Administrative record type code	BPv6 Administrative Record Type Code	1	“UNUSED_ZERO”*	“BUNDLE_STATUS_REPORT”	“SAGA_MESSAGE”	N/A
m_isFragment	Is fragment flag	bool	1	FALSE*	FALSE	TRUE	N/A

6.11.2 BPv6 Administrative Record Flags

Table 6-19: BPv6 Administrative Record Flags defines the BPv6 Administrative Record Flags bit-packed 8-bit structure.

Table 6-19: BPv6 Administrative Record Flags

Mnemonic	Bit Position	Description
spare	0	Reserved for future use
BUNDLE_IS_A_FRAGMENT	1	Bundle is a fragment
Spare	2-7	Reserved for future use

6.11.3 BPv6 Block Processing Flags

Table 6-20: BPv6 Block Processing Flags defines the BPv6 Block Processing Flags bit-packed 64-bit structure.

Table 6-20: BPv6 Block Processing Flags

Mnemonic	Bit Position	Description
MUST_BE_REPLICATED_IN_EVERY_FRAGMENT	0	Block must be replicated in every fragment
STATUS_REPORT_REQUESTED_IF_BLOCK_CANT_BE_PROCESSED	1	Transmit status report if block can't be processed
DELETE_BUNDLE_IF_BLOCK_CANT_BE_PROCESSED	2	Delete bundle if block can't be processed
IS_LAST_BLOCK	3	Last block
DISCARD_BLOCK_IF_IT_CANT_BE_PROCESSED	4	Discard block if it can't be processed
BLOCK_WAS_FORWARDED_WITHOUT_BEING_PROCESSED	5	Block was forwarded without being processed
BLOCK_CONTAINS_AN_EID_REFERENCE_FIELD	6	Block contains an EID-reference field
Spare	7-63	Reserved for future use

6.11.4 BPv6 Bundle Age Canonical Block

Table 6-21: BPv6 Bundle Age Canonical Block defines the BPv6 Bundle Age Block Canonical structure.

Table 6-21: BPv6 Bundle Age Canonical Block

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
Bpv6CanonicalBlock	BPv6 canonical block	BPv6 Canonical Block	N/A	N/A	N/A	N/A	N/A
largestSerializedDataOnlySize	Largest serialized data only size	uint64	8	10	10	10	bytes
m_bundleAgeMicroseconds	Bundle age microseconds	uint64	8	0*	UINT64_MIN	UINT64_MAX	µsec

6.11.5 BPv6 Bundle Processing Flags

Table 6-22: BPv6 Bundle Processing Flags defines the BPv6 Bundle Processing Flags bit-packed 64-bit structure.

Table 6-22: BPv6 Bundle Processing Flags

Mnemonic	Bit Position	Description
ISFRAGMENT	0	Bundle is a fragment
ADMINRECORD	1	Application data unit is an administrative record
NOFRAGMENT	2	Bundle must not be fragmented
CUSTODY_REQUESTED	3	Custody transfer is requested
SINGLETON	4	Destination endpoint is a singleton
USER_APP_ACK_REQUESTED	5	Acknowledgement by application is requested
Spare	6	Reserved for future use
PRIORITY_BIT_MASK	7-8	Bundle priority; BPv6 priority enumerated value
Spare	9-13	Reserved for future use
RECEPTION_STATUS_REPORTS_REQUESTED	14	Reporting of bundle reception is requested
CUSTODY_STATUS_REPORTS_REQUESTED	15	Reporting of custody acceptance is requested
FORWARDING_STATUS_REPORTS_REQUESTED	16	Reporting of bundle forwarding is requested
DELIVERY_STATUS_REPORTS_REQUESTED	17	Reporting of bundle delivery is requested
DELETION_STATUS_REPORTS_REQUESTED	18	Reporting of bundle deletion is requested
Spare	19-63	Reserved for future use

6.11.6 BPv6 Bundle Status Report Flags

Table 6-23: BPv6 Bundle Status Report Flags defines the BPv6 Bundle Status Report Flags bit-packed 8-bit structure.

Table 6-23: BPv6 Bundle Status Report Flags

Mnemonic	Bit Position	Description
REPORTING_NODE_RECEIVED_BUNDLE	0	Reporting node received bundle
REPORTING_NODE_ACCEPTED_CUSTODY_OF_BUNDLE	1	Reporting node accepted custody of bundle
REPORTING_NODE_FORWARDED_BUNDLE	2	Reporting node forwarded the bundle
REPORTING_NODE_DELIVERED_BUNDLE	3	Reporting node delivered the bundle
REPORTING_NODE_DELETED_BUNDLE	4	Reporting node deleted the bundle
Spare	5-7	Reserved for future use

6.11.7 BPv6 Canonical Block

Table 6-24: BPv6 Canonical Block defines the BPv6 Canonical Block structure.

Table 6-24: BPv6 Canonical Block

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_blockProcessingControlFlags	Block processing control flags	BPv6 Block Processing Flags	8	0x0000 0000 0000 0000*	N/A	N/A	N/A
m_blockTypeSpecificDataLength	Block type specific data length	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_blockTypeSpecificDataPtr	Block type specific data pointer	uint8 pointer	N/A	*	N/A	N/A	N/A
m_blockTypeCode	Block type code	BPv6 Block Type	1	"PRIMARY_IMPLICIT_ZERO"*	"PAYLOAD"	"BUNDLE_AGE"	N/A

6.11.8 BPv6 CBHE Primary Block

Table 6-25: BPv6 CBHE Primary Block defines the BPv6 CBHE Primary Block structure.

Table 6-25: BPv6 CBHE Primary Block

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_bundleProcessingControlFlags	Bundle processing control flags	BPv6 Bundle Processing Flags	8	0x0000 0000 0000 0000*	N/A	N/A	N/A
m_blockLength	Block length	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_destinationEid	Destination EID	CBHE EID	16	*	N/A	N/A	N/A
m_sourceNodeId	Source node ID	CBHE EID	16	*	N/A	N/A	N/A
m_reportToEid	Report to EID	CBHE EID	16	*	N/A	N/A	N/A
m_custodianEid	Custodian EID	CBHE EID	16	*	N/A	N/A	N/A
m_creationTimestamp	Creation timestamp	Timestamp Utility BPv6 Creation Timestamp	16	*	N/A	N/A	sec
m_lifetimeSeconds	Lifetime seconds	uint64	8	0*	UINT64_MIN	UINT64_MAX	sec
m_tmpDictionaryLengthIgnoredAndAssumedZero	Temporary dictionary length ignored and assumed zero	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_fragmentOffset	Fragment offset	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalApplicationDataUnitLength	Total application data unit length	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes

6.11.9 BPv6 Metadata Canonical Block

Table 6-26: BPv6 Metadata Canonical Block defines the BPv6 Metadata Canonical Block structure.

Table 6-26: BPv6 Metadata Canonical Block

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
Bpv6CanonicalBlock	BPv6 canonical block	BPv6 Canonical Block	N/A	*	N/A	N/A	N/A
m_metadataTypeCode	Metadata type code	BPv6 Metadata Type Code	8	"UNDEFINED_ZERO"*	"URI"	"URI"	N/A
m_metadataContentPtr	Metadata content pointer	BPv6 Metadata Content Base pointer	N/A	*	N/A	N/A	N/A

6.11.10 BPv6 Metadata Content Generic

Table 6-27: BPv6 Metadata Content Generic defines the BPv6 Metadata Content Generic structure.

Table 6-27: BPv6 Metadata Content Generic

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_genericRawMetadata	Generic raw metadata	uint8 vector	variable	*	N/A	N/A	N/A

6.11.11 BPv6 Metadata Content Uniform Resource Identifier List

Table 6-28: BPv6 Metadata Content URI List defines the BPv6 Metadata Content Uniform Resource Identifier (URI) List structure.

Table 6-28: BPv6 Metadata Content URI List

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_uriArray	URI array	CBHE EID vector	variable	*	N/A	N/A	N/A

6.11.12 BPv6 Previous Hop Insertion Canonical Block

Table 6-29: BPv6 Previous Hop Insertion Canonical Block defines the BPv6 Previous Hop Insertion Canonical Block structure.

Table 6-29: BPv6 Previous Hop Insertion Canonical Block

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
Bpv6CanonicalBlock	BPv6 canonical block	BPv6 Canonical Block	N/A	N/A	N/A	N/A	N/A
largestSerializedDataOnlySize	Largest serialized data only size	uint64	8	46	46	46	bytes
m_previousNode	Previous node	CBHE EID	16	*	N/A	N/A	N/A

6.11.13 BPv6 Primary Block View

Table 6-30: BPv6 Primary Block View defines the BPv6 Primary Block View structure.

Table 6-30: BPv6 Primary Block View

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
header	Header	BPv6 CBHE Primary Block	16	N/A	N/A	N/A	N/A
actualSerializedPrimaryBlockPtr	Actual serialized primary block pointer	Boost ASIO Constant Buffer	variable	*	N/A	N/A	N/A
dirty	Dirty flag	bool	1	FALSE*	FALSE	TRUE	N/A

6.12 Bundle Protocol Version 7 Data Structures

This section defines data structures specific to BPv7 implementation.

6.12.1 BPSec BCB-AES-GCM Additional Authentication Data Scope Mask

Table 6-31: BPSec BCB-AES-GCM Additional Authentication Data Scope Mask defines the BPSec BCB-AES-GCM Additional Authentication Data Scope Mask bit-packed 64-bit structure.

Table 6-31: BPSec BCB-AES-GCM Additional Authentication Data Scope Mask

Mnemonic	Bit Position	Description
INCLUDE_PRIMARY_BLOCK	0	Include primary block flag
INCLUDE_TARGET_HEADER	1	Include target header flag
INCLUDE_SECURITY_HEADER	2	Include security header flag
Spare	3-7	Reserved
Spare	8-63	Unassigned

6.12.2 BPSec BIB-HMAC-SHA2 Integrity Scope Mask

Table 6-32: BPSec BIB-HMAC-SHA2 Integrity Scope Mask defines the BPSec BIB-HMAC-SHA2 Integrity Scope Mask bit-packed 64-bit structure.

Table 6-32: BPSec BIB-HMAC-SHA2 Integrity Scope Mask

Mnemonic	Bit Position	Description
INCLUDE_PRIMARY_BLOCK	0	Include primary block flag
INCLUDE_TARGET_HEADER	1	Include target header flag
INCLUDE_SECURITY_HEADER	2	Include security header flag
Spare	3-7	Reserved
Spare	8-63	Unassigned

6.12.3 BPv7 Abstract Security Block

Table 6-33: BPv7 Abstract Security Block defines the BPv7 Abstract Security Block structure.

Table 6-33: BPv7 Abstract Security Block

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
Bpv7CanonicalBlock	BPv7 canonical block	BPv7 Canonical Block	N/A	N/A	N/A	N/A	N/A

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_securityTargets	Security targets	uint64 vector	variable	*	N/A	N/A	N/A
m_securityContextId	Security context ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_securityContextFlags	Security context flags	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
m_securitySource	Security source	CBHE EID	16	*	N/A	N/A	N/A
m_securityContextParametersOptional	Security context parameters optional	vector<pair<uint64, unique_ptr<Bpv7AbstractSecurityBlockValueBase>>>	variable	*	N/A	N/A	N/A
m_securityResults	Security results	vector<pair<uint64, unique_ptr<Bpv7AbstractSecurityBlockValueBase>>>	variable	*	N/A	N/A	N/A

6.12.4 BPv7 Abstract Security Block Value (byte string)

Table 6-34: BPv7 Abstract Security Block Value (byte string) defines the BPv7 Abstract Security Block Value (byte string) structure.

Table 6-34: BPv7 Abstract Security Block Value (byte string)

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_byteString	Byte string	uint8 vector	variable	*	N/A	N/A	N/A

6.12.5 BPv7 Abstract Security Block Value (uint)

Table 6-35: BPv7 Abstract Security Block Value (uint) defines the BPv7 Abstract Security Block Value (uint) structure.

Table 6-35: BPv7 Abstract Security Block Value (uint)

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_uintValue	Uint value	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.12.6 BPv7 Administrative Record

Table 6-36: BPv7 Administrative Record defines the BPv7 Administrative Record structure.

Table 6-36: BPv7 Administrative Record

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
Bpv7CanonicalBlock	BPv7 canonical block	BPv7 Canonical Block	N/A	N/A*	N/A	N/A	N/A
m_adminRecordTypeCode	Administrative record type code	BPv7 Administrative Record Type Code	8	"UNUSED_ZERO"*	"BUNDLE_STATUS_REPORT"	"CUSTODY_SIGNAL"	N/A
m_adminRecordContentPtr	Administrative record content pointer	BPv7 Administrative Record Content Base pointer	N/A	*	N/A	N/A	N/A

6.12.7 BPv7 Administrative Record Content Bundle Status Report

Table 6-37: BPv7 Administrative Record Content Bundle Status Report Block defines the BPv7 Administrative Record Content Bundle Status Report structure.

Table 6-37: BPv7 Administrative Record Content Bundle Status Report Block

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_bundleStatusInfo	Bundle status information	BPv7 Bundle Status Information Array	36	*	N/A	N/A	N/A
m_statusReportReasonCode	Status report reason code	BPv7 Bundle Status Report Reason Codes	8	"NO_FURTHER_INFORMATION"*	"NO_FURTHER_INFORMATION"	"BLOCK_UNSUPPORTED"	N/A
m_sourceNodeEid	Source node EID	CBHE EID	16	*	N/A	N/A	N/A
m_creationTimestamp	Creation timestamp	Timestamp Utility BPv7 Creation Timestamp	20	*	N/A	N/A	msec
m_optionalSubjectPayloadFragmentOffset	Optional subject payload fragment offset	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_optionalSubjectPayloadFragmentLength	Optional subject payload fragment length	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_subjectBundleIsFragment	Subject bundle is fragment flag	bool	1	FALSE*	FALSE	TRUE	N/A
m_reportStatusTimeFlagWasSet	Report status time flag was set flag	bool	1	FALSE*	FALSE	TRUE	N/A

6.12.8 BPv7 Administrative Record Content Bundle-in-Bundle Encapsulation Protocol Data Unit Message

Table 6-38: BPv7 Administrative Record Content BIBE PDU Message defines the BPv7 Administrative Record Content Bundle-in-Bundle Encapsulation (BIBE) Protocol Data Unit (PDU) Message structure.

Table 6-38: BPv7 Administrative Record Content BIBE PDU Message

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_transmissionId	Transmission ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_custodyRetransmissionTime	Custody retransmission time	uint64	8	0*	UINT64_MIN	UINT64_MAX	msec
m_encapsulatedBundlePtr	Encapsulated bundle pointer	uint8 pointer	N/A	*	N/A	N/A	N/A
m_encapsulatedBundleLength	Encapsulated bundle length	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_temporaryEncapsulatedBundleStorage	Temporary encapsulated bundle storage	uint8 vector	variable	*	N/A	N/A	N/A

6.12.9 BPv7 Block Processing Flags

Table 6-39: BPv7 Block Processing Flags defines the BPv7 Block Processing Flags bit-packed 64-bit structure.

Table 6-39: BPv7 Block Processing Flags

Mnemonic	Bit Position	Description
MUST_BE_REPLICATED	0	Block must be replicated in every fragment
STATUS_REPORT_REQUESTED_IF_BLOCK_CANT_BE_PROCESSED	1	Transmit status report if block can't be processed
DELETE_BUNDLE_IF_BLOCK_CANT_BE_PROCESSED	2	Delete bundle if block can't be processed
Spare	3	Reserved
REMOVE_BLOCK_IF_IT_CANT_BE_PROCESSED	4	Discard block if it can't be processed
Spare	5-6	Reserved
Spare	7-63	Unassigned

6.12.10 BPv7 Bundle Age Canonical Block

Table 6-40: BPv7 Bundle Age Canonical Block defines the BPv7 Bundle Age Canonical Block structure.

Table 6-40: BPv7 Bundle Age Canonical Block

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
Bpv7CanonicalBlock	BPv7 canonical block	BPv7 Canonical Block	N/A	N/A	N/A	N/A	N/A
largestSerializedDataOnlySize	Largest serialized data only size	uint64	8	9	9	9	bytes
m_bundleAgeMilliseconds	Bundle age milliseconds	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	msec

6.12.11 BPv7 Bundle Processing Flags

Table 6-41: BPv7 Bundle Processing Flags defines the BPv7 Bundle Processing Flags bit-packed 64-bit structure.

Table 6-41: BPv7 Bundle Processing Flags

Mnemonic	Bit Position	Description
ISFRAGMENT	0	The bundle is a fragment
ADMINRECORD	1	The bundle's payload is an administrative record
NOFRAGMENT	2	The bundle must not be fragmented
spare	3-4	Reserved
USER_APP_ACK_REQUESTED	5	Acknowledgment by the user application is requested
STATUSTIME_REQUESTED	6	Status time is requested in all status reports
spare	7-13	Reserved
RECEPTION_STATUS_REPORTS_REQUESTED	14	Reporting of bundle reception is requested
Spare	15	Reserved
FORWARDING_STATUS_REPORTS_REQUESTED	16	Reporting of bundle forwarding is requested
DELIVERY_STATUS_REPORTS_REQUESTED	17	Reporting of bundle delivery is requested
DELETION_STATUS_REPORTS_REQUESTED	18	Reporting of bundle deletion is requested
Spare	19-63	Reserved

6.12.12 BPv7 Canonical Block

Table 6-42: BPv7 Canonical Block defines the BPv7 Canonical Block structure.

Table 6-42: BPv7 Canonical Block

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
smallestSerializedCanonicalSize	Smallest serialized canonical size	uint64	8	6	6	6	bytes
largestZeroDataSerializedCanonicalSize	Largest zero data serialized canonical size	uint64	8	37	37	37	bytes
m_blockNumber	block number	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_blockProcessingControlFlags	Block processing control flags	BPv7 Block Processing Flags	8	0x0000 0000 0000 0000*	N/A	N/A	N/A
m_dataPtr	Data pointer	uint8 pointer	N/A	*	N/A	N/A	N/A
m_dataLength	Data length	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_computedCrc32	Computed CRC32	uint32	4	0*	UINT32_MIN	UINT32_MAX	N/A
m_computedCrc16	Computed CRC16	uint16	2	0*	UINT16_MIN	UINT16_MAX	N/A
m_blockTypeCode	Block type code	BPv7 Block Type Code	1	"PRIMARY_IMPLICIT_ZERO"*	"PAYLOAD"	"PRIORITY"	N/A
m_crcType	CRC type	BPv7 CRC Type	1	"NONE"*	"NONE"	"CRC32C"	N/A

6.12.13 BPv7 CBHE Primary Block

Table 6-43: BPv7 CBHE Primary Block defines the BPv7 Compressed Bundle Header Encoding (CBHE) Primary Block structure.

Table 6-43: BPv7 CBHE Primary Block

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
smallestSerializedPrimarySize	Smallest serial primary size	uint64	8	17	17	17	bytes
largestSerializedPrimarySize	Largest serialized primary size	uint64	8	120	120	120	bytes
m_bundleProcessingControlFlags	Bundle processing control flags	BPv7 Bundle Processing Flags	8	0x0000 0000 0000 0000*	N/A	N/A	N/A
m_destinationEid	Destination EID	CBHE EID	16	*	N/A	N/A	N/A
m_sourceNodeId	Source node ID	CBHE EID	16	*	N/A	N/A	N/A
m_reportToEid	Report to EID	CBHE EID	16	*	N/A	N/A	N/A
m_creationTimestamp	Creation timestamp	Timestamp Utility BPv7 Creation Timestamp	20	*	N/A	N/A	msec
m_lifetimeMilliseconds	Lifetime milliseconds	uint64	8	0*	UINT64_MIN	UINT64_MAX	msec
m_fragmentOffset	Fragment offset	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_totalApplicationDataUnitLength	Total application data unit length	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_computedCrc32	Computed CRC32	uint32	4	0*	UINT32_MIN	UINT32_MAX	N/A
m_computedCrc16	Computer CRC16	uint16	2	0*	UINT16_MIN	UINT16_MAX	N/A
m_crcType	CRC type	BPv7 CRC Type	1	"NONE"*	"NONE"	"CRC32C"	N/A

6.12.14 BPv7 Hop Count Canonical Block

Table 6-44: BPv7 Hop Count Canonical Block defines the BPv7 Hop Count Canonical Block structure.

Table 6-44: BPv7 Hop Count Canonical Block

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
Bpv7CanonicalBlock	BPv7 canonical block	BPv7 Canonical Block	N/A	N/A	N/A	N/A	N/A
largestSerializedDataOnlySize	Largest serialized data only size	uint64	8	19	19	19	bytes
m_hopLimit	Hop limit	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_hopCount	Hop count	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.12.15 BPv7 Previous Node Canonical Block

Table 6-45: BPv7 Previous Node Canonical Block defines the BPv7 Previous Node Canonical Block structure.

Table 6-45: BPv7 Previous Node Canonical Block

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
Bpv7CanonicalBlock	BPv7 canonical block	BPv7 Canonical Block	N/A	N/A	N/A	N/A	N/A
largestSerializedDataOnlySize	Largest serialized data only size	uint64	8	19	19	19	bytes
m_previousNode	Previous node	CBHE EID	16	*	N/A	N/A	N/A

6.12.16 BPv7 Primary Block View

Table 6-46: BPv7 Primary Block View defines the BPv7 Primary Block View structure.

Table 6-46: BPv7 Primary Block View

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
header	Header	BPv7 CBHE Primary Block	123	N/A	N/A	N/A	N/A
actualSerializedPrimaryBlockPtr	Actual serialized primary block pointer	Boost ASIO Constant Buffer	N/A	*	N/A	N/A	N/A
dirty	Dirty flag	bool	1	FALSE*	FALSE	TRUE	N/A

6.12.17 BPv7 Priority Canonical Block

Table 6-47: BPv7 Priority Canonical Block defines the BPv7 Priority Canonical Block structure.

Table 6-47: BPv7 Priority Canonical Block

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
Bpv7CanonicalBlock	BPv7 canonical block	BPv7 Canonical Block	N/A	N/A	N/A	N/A	N/A
largestSerializedDataOnlySize	Largest serialized data only size	uint64	8	9	9	9	bytes
m_bundlePriority	Bundle priority	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.13 Bundle Storage Catalog Entry Data Structures

This section defines data structures specific to Bundle Storage Catalog Entry implementation.

6.13.1 Bundle Storage Catalog Entry

Table 6-48: Bundle Storage Catalog Entry defines the Bundle Storage Catalog Entry structure.

Table 6-48: Bundle Storage Catalog Entry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
bundleSizeBytes	Bundle size in bytes	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
segmentIdChainVec	Segment ID chain vector	Segment ID Chain Vector	variable	*	N/A	N/A	N/A
destEid	Destination EID	CBHE EID	16	*	N/A	N/A	N/A
encodedAbsExpirationAndCustodyAndPriority	Encoded ABS expiration and custody and priority	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
sequence	Sequence	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
ptrUuidKeyInMap	UUID key in map	void pointer	N/A	*	N/A	N/A	N/A

6.14 Bundle Storage Manager Base Data Structures

This section defines data structures specific to Bundle Storage Manager Base implementation.

6.14.1 Bundle Storage Manager Session Read from Disk

Table 6-49: Bundle Storage Manager Session Read from Disk defines the Bundle Storage Manager Session Read from Disk structure.

Table 6-49: Bundle Storage Manager Session Read from Disk

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
catalogEntryPtr	Catalog entry pointer	Bundle Storage Catalog Entry pointer	N/A	*	N/A	N/A	N/A
custodyId	Custody ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
nextLogicalSegment	Next logical segment	uint32	4	0*	UINT32_MIN	UINT32_MAX	N/A

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
nextLogicalSegment ToCache	Next logical segment to Cache	uint32	4	0*	UINT32_ MIN	UINT32_ MAX	N/A
cacheReadIndex	Cache read index	uint32	4	0*	UINT32_ MIN	UINT32_ MAX	N/A
cacheWriteIndex	Cache write index	uint32	4	0*	UINT32_ MIN	UINT32_ MAX	N/A
readCache	Read cache	volatile uint8 array pointer	N/A	*	N/A	N/A	N/A

6.14.2 Bundle Storage Manager Session Write to Disk

Table 6-50: Bundle Storage Manager Session Write to Disk defines the Bundle Storage Manager Session Write to Disk structure.

Table 6-50: Bundle Storage Manager Session Write to Disk

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
catalogEntry	Catalog entry	Bundle Storage Catalog Entry	N/A	N/A	N/A	N/A	N/A
nextLogicalSegment	Next logical segment	uint32	4	0*	UINT32_ MIN	UINT32_ MAX	N/A

6.15 Compressed Bundle Header Encoding Data Structures

The following section defines data structures specific to CBHE implementation.

6.15.1 CBHE Bundle UUID

Table 6-51: CBHE Bundle UUID defines the CBHE Bundle UUID structure.

Table 6-51: CBHE Bundle UUID

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
creationSeconds	Creation seconds	uint64	8	0*	UINT64_MIN	UINT64_MAX	sec
sequence	Sequence	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
srcEid	Source EID	CBHE EID	16	*	N/A	N/A	N/A
fragmentOffset	Fragment offset	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
dataLength	Data length	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes

6.15.2 CBHE Bundle Universally Unique Identifier – No Fragment

Table 6-52: CBHE Bundle UUID – No Fragment defines the CBHE Bundle UUID – No Fragment structure.

Table 6-52: CBHE Bundle UUID – No Fragment

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
creationSeconds	Creation seconds	uint64	8	0*	UINT64_MIN	UINT64_MAX	sec
sequence	Sequence	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
srcEid	Source EID	CBHE EID	16	*	N/A	N/A	N/A

6.15.3 CBHE Endpoint Identifier

Table 6-53: CBHE EID defines the CBHE EID structure.

Table 6-53: CBHE EID

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
nodeId	Node ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
serviceId	Service ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.16 Configuration Data Structures

The following section defines data structures specific to Configuration implementation.

6.16.1 BPSec Security Failure Processing Action Masks

Table 6-54: BPSec Failure Processing Action Masks defines the BPSec Failure Processing Action Masks bit-packed 16-bit structure.

Table 6-54: BPSec Failure Processing Action Masks

Mnemonic	Bit Position	Description
REMOVE_SECURITY_OPERATION	0	Remove security operation
REMOVE_SECURITY_OPERATION_TARGET_BLOCK	1	Remove security operation target block
REMOVE_ALL_SECURITY_TARGET_OPERATIONS	2	Remove all security target operations
FAIL_BUNDLE_FORWARDING	3	Fail bundle forwarding
REQUEST_BUNDLE_STORAGE	4	Request bundle storage
REPORT_REASON_CODE	5	Report reason code
OVERRIDE_SECURITY_TARGET_BLOCK_BPCF	6	Override security target block Block Processing Control Flags (BPCF)
OVERRIDE_SECURITY_BLOCK_BPCF	7	Override security block BPCF
spare	8-15	unassigned

6.16.2 Policy Rules

Table 6-55: Policy Rules defines the Policy Rules structure.

Table 6-55: Policy Rules

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_description	Description	string	variable	“”*	N/A	N/A	N/A
m_securityPolicyRuleId	Security policy rule ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_securityRole	Security role	string	variable	“”*	N/A	N/A	N/A
m_securitySource	Security source	string	variable	“”*	N/A	N/A	N/A
m_bundleSource	Bundle source	string set	variable	*	N/A	N/A	N/A
m_bundleFinalDestination	Bundle final destination	string set	variable	*	N/A	N/A	N/A
m_securityTargetBlockTypes	Security target block types	uint64 set	variable	*	N/A	N/A	N/A
m_securityService	Security service	string	variable	“”*	N/A	N/A	N/A
m_securityContext	Security context	string	variable	“”*	N/A	N/A	N/A
m_securityFailureEventSetReferenceName	Security failure event set reference name	string	variable	“”*	N/A	N/A	N/A

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_securityFailureEventSetReferencePtr	Security failure event set reference pointer	Security Failure Event Sets pointer	N/A	*	N/A	N/A	N/A
m_securityContextParamsVec	Security context parameters vector	Security Context Parameters Vector	variable	*	N/A	N/A	N/A

6.16.3 Security Context Parameter

Table 6-56: Security Context Parameter defines the Security Context Parameter structure.

Table 6-56: Security Context Parameter

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_paramName	Parameter name	BPSec Security Context Parameter Name	1	*	N/A	N/A	N/A
m_valueUint	Unsigned integer value	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_valuePath	Path value	Boost Filesystem Path	variable	*	N/A	N/A	N/A

6.16.4 Security Failure Event Sets

Table 6-57: Security Failure Event Sets defines the Security Failure Event Sets structure.

Table 6-57: Security Failure Event Sets

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_name	Name	string	variable	“”*	N/A	N/A	N/A
m_description	Description	string	variable	“”*	N/A	N/A	N/A
m_securityOperationEventsVec	Security operation events vector	Security Operation Event Plus Actions Pairs Vector	variable	*	N/A	N/A	N/A
m_eventTypeToEventSetPtrLut	Event type to event set pointer lookup table	Event Type to Event Set Pointer Lookup Table	N/A	*	N/A	N/A	N/A

6.16.5 Security Operation Event Plus Actions Pair

Table 6-58: Security Operation Event Plus Actions Pair defines the Security Operation Event Plus Actions Pair structure.

Table 6-58: Security Operation Event Plus Actions Pair

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_event	Event	BPSec Security Failure Event	1	“UNDEFINED”*	“SECURITY_OPERATION_MISCONFIGURED_AT_VERIFIER”	“SECURITY_OPERATION_CORRUPTED_AT_ACCEPTOR”	N/A
m_actionMasks	Action masks	BPSec Security Failure Processing Action Mask	2	*	N/A	N/A	N/A

6.17 Contact Graph Routing Data Structures

The following section defines data structures specific to CGR implementation.

6.17.1 CGR CMR Map Data

Table 6-59: CGR CMR Map Data defines the CGR CMR Map Data structure.

Table 6-59: CGR CMR Map Data

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_vertex	Vertex	vertex	N/A	*	N/A	N/A	N/A
m_visited	Visited	bool	1	FALSE*	FALSE	TRUE	N/A
m_predecessorNodeId	Predecessor node ID	Node ID	8	*	N/A	N/A	N/A
m_arrivalTime	Arrival time	time	4	*	N/A	N/A	sec

6.18 DTN Real-Time Protocol Frame Data Structures

The following section defines data structures specific to DTN RTP Frame implementation.

6.18.1 Extension Header

Table 6-60: Extension Header defines the Extension Header 14-byte bit-packed structure.

Table 6-60: Extension Header

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Octet 0	type																len															
Octet 4	data																															
…																																
Octet N																																

Type is an unsigned 16-bit integer value indicating the profile-specific data type identifier. This value is initialized to 0.

Len is an unsigned 16-bit integer value indicating the length of the extension in 32-bit units, excluding the first 32-bits of the extension. This value is initialized to 0.

Data is the extension data, equal in length to 4 bytes times the value indicated by “len”. This data is referenced by pointer, and the pointer is initialized to “null”.

6.18.2 RTCP Application Packet

Table 6-61: RTCP Application Packet defines the RTCP Application Packet structure.

Table 6-61: RTCP Application Packet

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
header	Header	RTCP Header	6	N/A	N/A	N/A	N/A
ssrc	Synchronization source	uint32	4	0	UINT32_MIN	UINT32_MAX	N/A
name	Name	4-member uint8 array	4	{0}	N/A	N/A	N/A
payload	Payload	uint8 pointer	N/A	null	N/A	N/A	N/A

6.18.3 RTCP Header

Table 6-62: RTCP Header defines the RTCP Header structure.

Table 6-62: RTCP Header

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
version	Version	uint8	1	0	UINT8_MIN	UINT8_MAX	N/A
padding	Padding	uint8	1	0	UINT8_MIN	UINT8_MAX	N/A
count	Count	uint8	1	0	UINT8_MIN	UINT8_MAX	N/A
pkt_subtype	Packet subtype	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
pkt_type	Packet type	uint8	1	0	UINT8_MIN	UINT8_MAX	N/A
length	Length	uint8	1	0	UINT8_MIN	UINT8_MAX	words

6.18.4 RTCP Receiver Report

Table 6-63: RTCP Receiver Report defines the RTCP Receiver Report structure.

Table 6-63: RTCP Receiver Report

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
header	Header	RTCP Header	6	N/A	N/A	N/A	N/A
ssrc	Synchronization source	uint32	4	0	UINT32_MIN	UINT32_MAX	N/A
report_blocks	Report blocks	RTCP Report Block vector	variable	*	N/A	N/A	N/A

6.18.5 RTCP Report Block

Table 6-64: RTCP Report Block defines the RTCP Report Block structure.

Table 6-64: RTCP Report Block

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
ssrc	Synchronization source	uint32	4	0	UINT32_MIN	UINT32_MAX	N/A
fraction	Fraction	uint8	1	0	UINT8_MIN	UINT8_MAX	N/A
lost	Lost	int32	4	0	INT32_MIN	INT32_MAX	N/A
last_seq	Last sequence	uint32	4	0	UINT32_MIN	UINT32_MAX	N/A
jitter	Jitter	uint32	4	0	UINT32_MIN	UINT32_MAX	RTP timestamp units
lsr	Last sender report	uint32	4	0	UINT32_MIN	UINT32_MAX	N/A
dlsr	Delay since last sender report	uint32	4	0	UINT32_MIN	UINT32_MAX	RTP timestamp units

NOTE: RTP timestamp units are an abstract representation of time based on the media's clock rate. The following formula defines the relationship between time in seconds and time in RTP timestamp units:

$$(time\ in\ seconds) = \frac{(time\ in\ RTP\ timestamp\ units)}{(media\ clock\ rate)}$$

6.18.6 RTCP Sender Info

Table 6-65: RTCP Sender Info defines the RTCP Sender Info structure.

Table 6-65: RTCP Sender Info

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
ntp_msw	NTP most significant word	uint32	4	0	UINT32_MIN	UINT32_MAX	N/A
ntp_lsw	NTP least significant word	uint32	4	0	UINT32_MIN	UINT32_MAX	N/A
rtp_ts	RTP timestamp	uint32	4	0	UINT32_MIN	UINT32_MAX	RTP timestamp units
pkt_cnt	Packet count	uint32	4	0	UINT32_MIN	UINT32_MAX	N/A
byte_cnt	Byte count	uint32	4	0	UINT32_MIN	UINT32_MAX	N/A

NOTE: RTP timestamp units are an abstract representation of time based on the media's clock rate. The following formula defines the relationship between time in seconds and time in RTP timestamp units:

$$(time\ in\ seconds) = \frac{(time\ in\ RTP\ timestamp\ units)}{(media\ clock\ rate)}$$

6.18.7 RTCP Sender Report

Table 6-66: RTCP Receiver Report defines the RTCP Sender Report structure.

Table 6-66: RTCP Receiver Report

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
header	Header	RTCP Header	6	N/A	N/A	N/A	N/A
ssrc	Synchronization source	uint32	4	0	UINT32_ MIN	UINT32_ MAX	N/A
sender_info	Sender info	RTCP Sender Info	20	N/A	N/A	N/A	N/A
report_blocks	Report blocks	RTCP Report Block vector	variable	*	N/A	N/A	N/A

6.18.8 RTCP Stream Identifier Source Description Chunk

Table 6-67: RTCP SDES Chunk defines the RTCP Stream Identifier Source Description (SDES) Chunk structure.

Table 6-67: RTCP SDES Chunk

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
ssrc	Synchronization source	uint32	4	0	UINT32_MIN	UINT32_MAX	N/A
items	Items	RTCP SDES Item vector	variable	*	N/A	N/A	N/A

6.18.9 RTCP Stream Identifier Source Description Item

Table 6-68: RTCP SDES Item defines the RTCP SDES Item structure.

Table 6-68: RTCP SDES Item

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
type	Type	uint8	1	0	UINT8_MIN	UINT8_MAX	N/A
length	Length	uint8	1	0	UINT8_MIN	UINT8_MAX	bytes
data	Data	uint8 pointer	N/A	null	N/A	N/A	N/A

6.18.10 RTCP Stream Identifier Source Description Packet

Table 6-69: RTCP SDES Packet defines the RTCP SDES Packet structure.

Table 6-69: RTCP SDES Packet

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
header	Header	RTCP Header	6	N/A	N/A	N/A	N/A
chunks	Chunks	RTCP SDES Chunk vector	variable	*	N/A	N/A	N/A

6.18.11 RTP Flags

Table 6-70: RTP Flags defines the RTP Flags bit-packed 16-bit structure.

Table 6-70: RTP Flags

Mnemonic	Bit Position	Description
RTP_VERSION_TWO_FLAG	0-1	Version two flag
RTP_PADDING_FLAG	2	Padding flag
RTP_EXT_FLAG	3	Extension flag
RTP_CSRC_FMASK	4-7	Contributing Source (CSRC) field mask
RTP_MARKER_FLAG	8	Marker flag
RTP_PAYLOAD_MASK	9-15	Payload mask

6.18.12 RTP Header

Table 6-71: RTP Header defines the RTP Header 12-byte bit-packed structure.

Table 6-71: RTP Header

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Octet 0	V			P	X	CC				M	PT						Sequence Number																
Octet 4	Timestamp																															
Octet 8	SSRC Identifier																															

Version (V) is a 2-bit field indicating the version of the protocol.

Padding (P) is a 1-bit flag indicating the absence or presence of padding bytes at the end of the RTP packet.

- FALSE = no padding present
- TRUE = padding present

When padding is present, the last byte of padding indicates the number of bytes that were added (including itself).

Extension (X) is a 1-bit flag indicating the absence or presence of an extension header between the header and payload data.

- FALSE = no extension header present
- TRUE = extension header present

CSRC Count (CC) is a 4-bit field indicating the number of CSRC identifiers that follow the SSRC.

Marker (M) is a 1-bit flag indicating whether the current data has special relevance to the application.

- FALSE = data has no special relevance
- TRUE = data has special relevance

Payload Type (PT) is a 7-bit field indicating the format of the payload. Valid values for PT are not defined.

Sequence Number is a 16-bit field containing the sequence number of the packet used to detect packet loss.
Timestamp is a 32-bit field indicating the time at which the received sample appears in the stream. This value is expressed as a measure of RTP timestamp units. RTP timestamp units are an abstract representation of time based on the media's clock rate. The following formula defines the relationship between time in seconds and time in RTP timestamp units:

$$(time\ in\ seconds) = \frac{(time\ in\ RTP\ timestamp\ units)}{(media\ clock\ rate)}$$

SSRC Identifier is a 32-bit field that uniquely identifies the source of a stream. The SSRC associated with each stream source is chosen at random at the start of each RTP session.

6.18.13 RTP Header Union

Table 6-72: RTP Header Union defines the RTP Header Union 14-byte bit-packed structure.

Table 6-72: RTP Header Union

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Octet 0	Flags																RTP Header (octets 0-1)																
Octet 4	RTP Header (octets 2-5)																															
Octet 8	RTP Header (octets 6-9)																															
Octet 12	RTP Header (octets 10-11)																															

Flags is an unsigned 16-bit integer value.

RTP Header is the 12-byte RTP Header structure.

6.18.14 RTP Frame

Table 6-73: RTP Frame defines the RTP Frame structure.

Table 6-73: RTP Frame

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
header	Header	RTP Header	12	N/A	N/A	N/A	N/A
payload	Payload	buffer	variable	*	N/A	N/A	N/A

6.19 DTN Utility Data Structures

The following section defines data structures specific to DTN Utility implementation.

6.19.1 Buffer

Table 6-74: Buffer defines the Buffer structure.

Table 6-74: Buffer

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
start	Start	void pointer	N/A	*	N/A	N/A	N/A
length	Length	size	8	*	N/A	N/A	bytes

6.20 Induct Configuration Data Structures

The following section defines data structures specific to Induct Configuration.

6.20.1 Induct Element Configuration

Table 6-75: Induct Element Configuration defines the Induct Element Configuration structure.

Table 6-75: Induct Element Configuration

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
name	Identifier provided by the user via the configuration file	string	variable	“”*	N/A	N/A	N/A
convergenceLayer	Convergence layer type. Possible values are: • stcp • tcpcl_v3 • tcpcl_v4 • udp • ltp_over_udp	string	variable	“”*	N/A	N/A	N/A
boundPort	Port number the convergence layer will be bound to	uint16	2	0*	UINT16_ MIN	UINT16_ MAX	N/A

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
numRxCircularBufferElements	Number of receiver circular buffer elements. • For STCP this represents the number of bundles to buffer up. • For TCPCL this is the number of bundles or data fragments. • For LTP this is the number of UDP packets. • For UDP this is the number of packets/bundles.	uint32	4	0*	UINT32_ MIN	UINT32_ MAX	N/A
numRxCircularBufferBytesPerElement	The maximum size, in bytes, of each element in the circular receive buffer; only used for UDP and TCPCL	uint32	4	0*	UINT32_ MIN	UINT32_ MAX	N/A
bpEncapLocalSocketOrPipePath	BP encapsulation local socket or pipe path	string	variable	“”*	N/A	N/A	N/A
thisLtpEngineId	This LTP engine ID	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
remoteLtpEngineId	Remote LTP engine ID	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
ltpReportSegmentMtu	MTU size for LTP report segments. Only used for the “ltp_over_udp” convergence layer.	uint32	4	0*	UINT32_ MIN	UINT32_ MAX	bytes
oneWayLightTimeMs	This is the one-way light time. Round trip time (retransmission time) is computed as (2 * (oneWayLightTime + oneWayMarginTime)) .	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	msec
oneWayMarginTimeMs	This is the one-way margin (packet processing) time. Round trip time (retransmission time) is computed by (2 * (oneWayLightTime + oneWayMarginTime)) .	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	msec
clientServiceId	Client service ID; only used for “ltp_over_udp” convergence layer.	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
preallocatedRedDataBytes	Number of red data bytes to preallocate on a receiver.	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
ltpMaxRetriesPerSerialNumber	Maximum number of retries/resends of a single LTP packet with a serial number before the session is terminated.	uint32	4	0*	UINT32_MIN	UINT32_MAX	N/A
ltpRandomNumberSizeBits	LTP random number size bits Number of bits to use for random numbers in LTP. Possible values are • 32 • 64	uint32	4	0*	UINT32_MIN	UINT32_MAX	bits
ltpEncapLocalSocketOrPipePath	LTP encapsulation local socket or pipe path	string	variable	“”*	N/A	N/A	N/A
ltpRemoteUdpHostname	LTP remote UDP hostname or IP address	string	variable	“”*	N/A	N/A	N/A
ltpRemoteUdpPort	LTP remote UDP port	uint16	2	0*	UINT16_MIN	UINT16_MAX	N/A
ltpRxDataSegmentSessionNumberRecreationPreventerHistorySize	Number of recent LTP receiver history session numbers to remember	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
ltpMaxExpectedSimultaneousSessions	LTP max number of expected simultaneous sessions	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
ltpMaxUdpPacketsToSendPerSystemCall	LTP max UDP packets to send per system call	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
delaySendingOfReportSegmentsTimeMsOrZeroToDisable	Time to defer data retransmission to efficiently handle out-of-order report segments; only used for “ltp_over_udp” convergence layer.	uint64	8	0*	UINT64_MIN	UINT64_MAX	msec
keepActiveSessionDataOnDisk	Current LTP session needs to be kept active on a solid-state disk drive flag	bool	1	FALSE*	FALSE	TRUE	N/A
activeSessionDataOnDiskNewFileDurationMs	The time window for which write allocations will be redirected to the currently active write file; only used if “keepActiveSessionDataOnDisk” is TRUE	uint64	8	0*	UINT64_MIN	UINT64_MAX	msec
activeSessionDataOnDiskDirectory	Working directory path; only used if “keepActiveSessionDataOnDisk” is TRUE	Boost Filesystem Path	variable	*	N/A	N/A	N/A
comPort	Com port	string	variable	“”*	N/A	N/A	N/A
baudRate	Baud rate	uint32	4	0*	UINT32_MIN	UINT32_MAX	symbols/sec
remoteNodeId	Remote node ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
keepAliveIntervalSeconds	Minimum interval to negotiate as the session keepalive	uint16	2	0*	UINT16_MIN	UINT16_MAX	sec
tcpclV3MyMaxTxSegmentSizeBytes	TCPCLV3 maximum segment size for transmitting data segments	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
tcpclV4MyMaxRxSegmentSizeBytes	TCPCLV4 maximum segment size for transmitting data segments	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
tlsIsRequired	TLS is required flag	bool	1	FALSE*	FALSE	TRUE	N/A
certificatePemFile	Path to the file containing the certificate for Secure Socket Layer (SSL)	Boost Filesystem Path	variable	*	N/A	N/A	N/A
privateKeyPemFile	Path to the file containing the private key for SSL	Boost Filesystem Path	variable	*	N/A	N/A	N/A
diffieHellmanParametersPemFile	Path to the file containing the Diffie-Hellman parameters for TLS	Boost Filesystem Path	variable	*	N/A	N/A	N/A

6.21 Initialization Vectors Data Structures

This section defines data structures specific to Initialization Vectors implementation.

6.21.1 Initialization Vector 12-Byte

Table 6-76: Initialization Vector 12-Byte defines the Initialization Vector 12-Byte structure.

Table 6-76: Initialization Vector 12-Byte

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_timePart	Time part	uint64	8	0*	UINT64_MIN	UINT64_MAX	µsec
m_counterPart	Counter part	uint32	4	0*	UINT32_MIN	UINT32_MAX	N/A

6.21.2 Initialization Vector 16-Byte

Table 6-77: Initialization Vector 16-Byte defines the Initialization Vector 16-Byte structure.

Table 6-77: Initialization Vector 16-Byte

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_timePart	Time part	uint64	8	0*	UINT64_MIN	UINT64_MAX	µsec
m_counterPart	Counter part	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.21.3 Initialization Vectors for One Thread

Table 6-78: Initialization Vectors for One Thread defines the Initialization Vectors for One Thread structure.

Table 6-78: Initialization Vectors for One Thread

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_iv12	Initialization vector 12-byte	Initialization Vector 12-Byte	12	*	N/A	N/A	N/A
m_iv16	Initialization vector 16-byte	Initialization Vector 16-Byte	16	*	N/A	N/A	N/A
m_initializationVector	Initialization vector	uint8 vector	variable	*	N/A	N/A	N/A

6.22 Licklider Transmission Protocol Client Service Data To Send Data Structures

The following section defines data structures specific to LTP Client Service Data To Send implementation.

6.22.1 UDP Send Packet Information

Table 6-79: UDP Send Packet Information defines the UDP Send Packet Information structure.

Table 6-79: UDP Send Packet Information

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
constBufferVec	Constant buffer vector	Boost ASIO constant buffer vector	variable	*	N/A	N/A	N/A
underlyingDataToDeleteOnSentCallback	Underlying data to delete on sent callback	uint8 vector vector shared pointer	N/A	*	N/A	N/A	N/A
underlyingCsDataToDeleteOnSentCallback	Underlying CS data to delete on sent callback	LTP Client Service Data To Send shared pointer	N/A	*	N/A	N/A	N/A
deferredRead	Deferred read	Deferred Read	N/A	*	N/A	N/A	N/A
sessionOriginatorEngineId	Session originator engine ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.23 Licklider Transmission Protocol Engine Configuration Data Structures

The following section defines data structures specific to LTP Engine Configuration implementation.

6.23.1 LTP Engine Configuration

Table 6-80: LTP Engine Configuration defines the LTP Engine Configuration structure.

Table 6-80: LTP Engine Configuration

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
thisEngineId	This engine ID	uint64	8	0	UINT64_MIN	UINT64_MAX	N/A
remoteEngineId	Remote engine ID	uint64	8	0	UINT64_MIN	UINT64_MAX	N/A
clientServiceId	Client service ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
isInduct	Is induct flag	bool	1	FALSE	FALSE	TRUE	N/A
mtuClientServiceData	MTU client service data	uint64	8	1360	UINT64_MIN	UINT64_MAX	bytes
mtuReportSegment	MTU report segment	uint64	8	1360	UINT64_MIN	UINT64_MAX	bytes
oneWayLightTime	One way light time	Boost Posix Time Duration	8	1000	N/A	N/A	msec
oneWayMarginTime	One way margin time	Boost Posix Time Duration	8	200	N/A	N/A	msec
remoteHostname	Remote hostname	string	variable	“localhost”	N/A	N/A	N/A
remotePort	Remote port	uint16	2	1113	UINT16_MIN	UINT16_MAX	N/A
myBoundUdpPort	My bound UDP port	uint16	2	1113	UINT16_MIN	UINT16_MAX	N/A
encapLocalSocketOrPipePath	Encapsulation local socket or pipe path	string	variable	“/tmp/ltp_local_socket”	N/A	N/A	N/A
numUdpRxCircularBufferVectors	Number of UDP receive circular buffer vectors	uint32	4	1000	UINT32_MIN	UINT32_MAX	N/A
estimatedBytesToReceivePerSession	Estimated bytes to receive per session	uint64	8	1000000	UINT64_MIN	UINT64_MAX	bytes
maxRedRxBytesPerSession	Maximum red received bytes per session	uint64	8	10000000	UINT64_MIN	UINT64_MAX	bytes
checkpointEveryNthDataPacketSender	Checkpoint every nth data packet sender	uint32	4	0	UINT32_MIN	UINT32_MAX	N/A
maxRetriesPerSerialNumber	Maximum retries per serial number	uint32	4	5	UINT32_MIN	UINT32_MAX	N/A
force32BitRandomNumbers	Force 32-bit random numbers flag	bool	1	FALSE	FALSE	TRUE	N/A
maxSendRateBitsPerSecOrZeroToDisable	Maximum send rate bits per second OR zero to disable	uint64	8	0	UINT64_MIN	UINT64_MAX	bits per sec

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
maxSimultaneousSessions	Maximum simultaneous sessions	uint64	8	5000	UINT64_MIN	UINT64_MAX	N/A
rxDataSegmentSessionNumberRecreationPreventerHistorySizeOrZeroToDisable	Receive data segment sessions number recreation preventer history OR zero to disable	uint64	8	0	UINT64_MIN	UINT64_MAX	N/A
maxUdpPacketsToSendPerSystemCall	maximum UDP packets to send per system call	uint64	8	1	UINT64_MIN	UINT64_MAX	N/A
senderPingSecondsOrZeroToDisable	Sender ping seconds OR zero to disable	uint64	8	0	UINT64_MIN	UINT64_MAX	sec
delaySendingOfReportSegmentsTimeMsOrZeroToDisable	delay sending of report segments time milliseconds OR zero to disable	uint64	8	20	UINT64_MIN	UINT64_MAX	msec
delaySendingOfDataSegmentsTimeMsOrZeroToDisable	Delay sending of data segments time milliseconds OR zero to disable	uint64	8	20	UINT64_MIN	UINT64_MAX	msec
activeSessionDataOnDiskNewFileDurationMsOrZeroToDisable	Active session data on disk new file duration milliseconds OR zero to disable	uint64	8	0	UINT64_MIN	UINT64_MAX	msec
activeSessionDataOnDiskDirectory	Active session data on disk directory	Boost Filesystem Path	variable	"./"	N/A	N/A	N/A
rateLimitPrecisionMicroSec	Rate limit precision microseconds	uint64	8	0	UINT64_MIN	UINT64_MAX	µsec

6.24 Message Data Structures

The following section defines data structures specific to Message implementation.

6.24.1 Common Header

Table 6-81: Common Header defines the Common Header structure.

Table 6-81: Common Header

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
type	Message type	uint16	2	0*	UINT16_MIN	UINT16_MAX	N/A
flags	Message flags	uint16	2	0*	UINT16_MIN	UINT16_MAX	N/A

6.24.2 Contact Plan Reload Header

Table 6-82: Contact Plan Reload Header defines the Contact Plan Reload Header structure.

Table 6-82: Contact Plan Reload Header

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
base	Base	Common Header	4	N/A	N/A	N/A	N/A
unusedPadding	Unused padding	4-member array of uint8	4	*	N/A	N/A	N/A

6.24.3 Depleted Storage Report Header

Table 6-83: Depleted Storage Report Header defines the Depleted Storage Report Header structure.

Table 6-83: Depleted Storage Report Header

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
base	Base	Common Header	4	N/A	N/A	N/A	N/A
nodeId	Node ID	uint64	8	0*	N/A	N/A	N/A

6.24.4 Egress Acknowledge Header

Table 6-84: Egress Acknowledge Header defines the Egress Acknowledge Header structure.

Table 6-84: Egress Acknowledge Header

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
base	Used to identify the type of message	Common Header	4	N/A	N/A	N/A	N/A
error	Type of egress errors	Egress Acknowledge Error Type	1	"NO_ ERRORS"*	"NO_ERRORS"	"NO_OUTDUCT"	N/A
deleteNow	Bundle can be deleted after sending when set to 1.	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
isResponseToStorageCutThrough	Bundle has already been placed in storage when set to 1; used between egress and storage	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
unused1	Unused	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
nextHopNodeId	Node ID for egress to send the bundle to next	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
finalDestEid	The node ID for the final destination of the bundle	CBHE EID	16	*	N/A	N/A	
custodyId	Custody ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
outductIndex	Outduct index	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.24.5 Link Status Header

Table 6-85: Link Status Header defines the Link Status Header structure.

Table 6-85: Link Status Header

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
base	Base	Common Header	4	N/A	N/A	N/A	N/A
event	Event	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
uuid	UUID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
unixTimeSecondsSince1970	Unix time in seconds since midnight (UTC), January 1, 1970	uint64	8	0*	UINT64_MIN	UINT64_MAX	sec

6.24.6 Release Change Header

Table 6-86: Release Change Header defines the Release Change Header structure.

Table 6-86: Release Change Header

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
subscriptionBytes	Subscription bytes	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
base	Base	Common Header	4	N/A	N/A	N/A	N/A
unused1	Unused byte 1	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
outductArrayIndex	Outduct array index	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
rateBps	Rate in bps	uint64	8	0*	UINT64_MIN	UINT64_MAX	bits per sec

6.24.7 Route Update Header

Table 6-87: Route Update Header defines the Route Update Header structure.

Table 6-87: Route Update Header

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
base	Base	Common Header	4	N/A	N/A	N/A	N/A
unused3	Unused byte 3	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
unused4	Unused byte 4	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
nextHopNodeId	Next hop node ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
finalDestNodeId	Final destination node ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.24.8 Schedule Header

Table 6-88: Schedule Header defines the Schedule Header structure.

Table 6-88: Schedule Header

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
base	Base	Common Header	4	N/A	N/A	N/A	N/A
flowId	Flow ID	uint32	4	0*	UINT32_MIN	UINT32_MAX	N/A
rate	Rate	uint64	8	0*	UINT64_MIN	UINT64_MAX	bits/sec
offset	Offset	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
duration	Duration	uint64	8	0*	UINT64_MIN	UINT64_MAX	sec

6.24.9 Storage Acknowledge Header

Table 6-89: Storage Acknowledge Header defines the Storage Acknowledge Header structure.

Table 6-89: Storage Acknowledge Header

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
base	Base	Common Header	4	N/A	N/A	N/A	N/A
error	Error	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
unused1	Unused byte 1	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
unused2	Unused byte 2	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
unused3	Unused byte 3	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
ingressUniqueId	Ingress unique ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
outductIndex	Outduct index	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.24.10 Telemetry Storage Header

Table 6-90: Telemetry Storage Header defines the Telemetry Storage Header structure.

Table 6-90: Telemetry Storage Header

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
base	Base	Common Header	4	N/A	N/A	N/A	N/A
stats	Statistics	Common Storage Statistics	120	N/A	N/A	N/A	N/A

6.24.11 To Egress Header

Table 6-91: To Egress Header defines the To Egress Header structure.

Table 6-91: To Egress Header

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
base	Base	Common Header	4	N/A	N/A	N/A	N/A
hasCustody	Has custody	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
isCutThroughFromStorage	Is cut through from storage	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
unused1	Unused byte 1	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
unused2	Unused byte 2	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
nextHopNodeId	Next hop node ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
finalDestEid	Final destination EID	CBHE EID	16	*	N/A	N/A	N/A

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
custodyId	Custody ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
outductIndex	Outduct index	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.24.12 To Storage Header

Table 6-92: To Storage Header defines the To Storage Header structure.

Table 6-92: To Storage Header

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
base	Base	Common Header	4	N/A	N/A	N/A	N/A
dontStoreBundle	Do not store bundle	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
isCustodyOrAdminRecord	Is custody or admin record	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
unused3	Unused byte 3	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
unused4	Unused byte 4	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A
ingressUniqueId	Ingress unique ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
outductIndex	Outduct index	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
finalDestEid	Final destination EID	CBHE EID	16	*	N/A	N/A	N/A

6.25 Outduct Data Structures

The following section defines data structures specific to Outduct implementation.

6.25.1 Outduct Element Configuration

Table 6-93: Outduct Element Configuration defines the Outduct Element Configuration structure.

Table 6-93: Outduct Element Configuration

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
name	Name	string	variable	“”*	N/A	N/A	N/A
convergenceLayer	Convergence layer	string	variable	“”*	N/A	N/A	N/A
nextHopNodeId	Next hop node ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
remoteHostname	Remote hostname	string	variable	“”*			N/A
remotePort	Remote port	uint16	2	0*	UINT16_MIN	UINT16_MAX	N/A
maxNumberOfBundlesInPipeline	Maximum number of bundles in pipeline	uint32	4	0*	UINT32_MIN	UINT32_MAX	N/A
maxSumOfBundleBytesInPipeline	Maximum sum of bundle bytes in pipeline	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
bpEncapLocalSocketOrPipePath	BP encapsulation local socket or pipe path	string	variable	“”*	N/A	N/A	N/A
thisLtpEngineId	This LTP engine ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
remoteLtpEngineId	Remote LTP engine ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
ltpDataSegmentMtu	LTP data segment MTU	uint32	4	0*	UINT32_MIN	UINT32_MAX	bytes

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
oneWayLightTimeMs	One way light time in milliseconds	uint64	8	0*	UINT64_MIN	UINT64_MAX	msec
oneWayMarginTimeMs	One way magin time in milliseconds	uint64	8	0*	UINT64_MIN	UINT64_MAX	msec
clientServiceId	Client service ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
numRxCircularBufferElements	Number of received circular buffer elements	uint32	4	0*	UINT32_MIN	UINT32_MAX	N/A
ltpMaxRetriesPerSerialNumber	LTP maximum retries per serial number	uint32	4	0*	UINT32_MIN	UINT32_MAX	N/A
ltpCheckpointEveryNthDataSegment	LTP checkpoint every nth data segment	uint32	4	0*	UINT32_MIN	UINT32_MAX	N/A
ltpRandomNumberSizeBits	LTP random number size in bits	uint32	4	0*	UINT32_MIN	UINT32_MAX	bits
ltpEncapLocalSocketOrPipePath	LTP encapsulation local socket or pipe path	string	variable	“”*	N/A	N/A	N/A
ltpSenderBoundPort	LTP sender bound port	uint16	2	0*	UINT16_MIN	UINT16_MAX	N/A
ltpMaxUdpPacketsToSendPerSystemCall	LTP maximum UDP packets to send per system call	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
ltpSenderPingSecondsOrZeroToDisable	LTP sender ping seconds or zero to disable	uint64	8	0*	UINT64_MIN	UINT64_MAX	sec
delaySendingOfDataSegmentsTimeMsOrZeroToDisable	Delay sending of data segments time in milliseconds or zero to disable	uint64	8	0*	UINT64_MIN	UINT64_MAX	msec
keepActiveSessionDataOnDisk	Keep active session data on disk	bool	1	FALSE*	FALSE	TRUE	N/A
activeSessionDataOnDiskNewFileDurationMs	Active session data on disk new file duration in milliseconds	uint64	8	0*	UINT64_MIN	UINT64_MAX	msec
activeSessionDataOnDiskDirectory	Active session data on disk directory	Boost Filesystem Path	variable	*	N/A	N/A	N/A
rateLimitPrecisionMicroSec	Rate limit precision in microseconds	uint64	8	0*	UINT64_MIN	UINT64_MAX	µsec
comport	Com port	string	variable	“”*	N/A	N/A	N/A
baudRate	Baud rate	uint32	4	0*	UINT32_MIN	UINT32_MAX	symbols/sec
keepAliveIntervalSeconds	Keep alive interval in seconds	uint16	2	0*	UINT16_MIN	UINT16_MAX	sec
tcpclV3MyMaxTxSegmentSizeBytes	TCPCLv3 my maximum transmit segment size in bytes	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
tcpclAllowOpportunisticReceiveBundles	TCPCL allow opportunistic receive bundles	bool	1	FALSE*	FALSE	TRUE	N/A
tcpclV4MyMaxRxSegmentSizeBytes	TCPCLv4 my maximum receive segment size in bytes	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
tryUseTls	Try use TLS	bool	1	FALSE*	FALSE	TRUE	N/A
tlsIsRequired	TLS is required	bool	1	FALSE*	FALSE	TRUE	N/A
useTlsVersion1_3	Use TLS version 1.3	bool	1	FALSE*	FALSE	TRUE	N/A
doX509CertificateVerification	Do X509 certificate verification	bool	1	FALSE*	FALSE	TRUE	N/A
verifySubjectAltNameInX509Certificate	Verify subject alternate name in X509 Certificate	bool	1	FALSE*	FALSE	TRUE	N/A
certificationAuthorityPemFileForVerification	Certification authority .PEM file for verification	Boost Filesystem Path	variable	*	N/A	N/A	N/A

6.25.2 Outduct Final Statistics

Table 6-94: Outduct Final Statistics defines the Outduct Final Statistics structure.

Table 6-94: Outduct Final Statistics

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_convergenceLayer	Convergence layer	string	variable	“”*	N/A	N/A	N/A
m_totalBundlesSent	Total bundles sent	size	8	*	N/A	N/A	N/A
m_totalBundlesAcked	Total bundles acknowledged	size	8	*	N/A	N/A	N/A

6.26 Serial Line Internet Protocol Data Structures

The following section defines data structures specific to SLIP implementation.

6.26.1 SLIP Decode State

Table 6-95: SLIP Decode State defines the SLIP Decode State structure.

Table 6-95: SLIP Decode State

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_previouslyReceivedChar	Previously received character	uint8	1	0*	UINT8_MIN	UINT8_MAX	N/A

6.27 Statistics Data Structures

The following section defines data structures specific to Statistics implementation.

6.27.1 Flow Statistics

Table 6-96: Flow Statistics defines the Flow Statistics structure.

Table 6-96: Flow Statistics

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
diskUsed	Disk storage used	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
diskWcount	Disk write operations	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
diskWbytes	Disk bytes written	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
diskRcount	Disk read operations	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
diskRbytes	Disk bytes read	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes

6.27.2 Storage Flow Statistics

Table 6-97: Storage Flow Statistics defines the Storage Flow Statistics structure.

Table 6-97: Storage Flow Statistics

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
src	Source flow label	uint32	4	0*	UINT32_MIN	UINT32_MAX	N/A
dst	Destination flow label	uint32	4	0*	UINT32_MIN	UINT32_MAX	N/A
rate	Maximum flow release rate	uint64	8	0*	UINT64_MIN	UINT64_MAX	bits/ sec
duration	Length of release time	uint64	8	0*	UINT64_MIN	UINT64_MAX	sec
start	Scheduled release time offset	uint64	8	0*	UINT64_MIN	UINT64_MAX	sec

6.27.3 Storage Statistics

Table 6-98: Storage Statistics defines the Storage Statistics structure.

Table 6-98: Storage Statistics

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
ts	Timestamp	double	8	0.0 *	0.0	DBL_MAX	sec
inMsg	Total message count in	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
inBytes	Total bytes in	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
outMsg	Total message count out	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
outBytes	Total bytes out	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
bytesUsed	Number of bytes used by storage	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
bytesAvailable	Number of bytes available to storage	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
rate	Current data release rate	uint64	8	0*	UINT64_MIN	UINT64_MAX	bits / sec
worker	Worker stats	Worker Statistics	56	N/A	N/A	N/A	N/A

6.27.4 Worker Statistics

Table 6-99: Worker Statistics defines the Worker Statistics structure.

Table 6-99: Worker Statistics

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
imsgSent	Messages sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
imsgReceived	Messages received	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
FlowStats	Flow stats	Flow Statistics	40	N/A	N/A	N/A	N/A

6.28 Storage Data Structures

The following section defines data structures specific to Storage implementation.

6.28.1 Storage Disk Configuration

Table 6-100: Storage Disk Configuration defines the Storage Disk Configuration structure.

Table 6-100: Storage Disk Configuration

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
name	Name	string	variable	“”*	N/A	N/A	N/A
storeFilePath	Storage file path	string	variable	“”*	N/A	N/A	N/A

6.29 Telemetry Definitions Data Structures

The following section defines data structures specific to Telemetry Definitions implementation.

6.29.1 All Induct Telemetry

Table 6-101: All Induct Telemetry defines the All Induct Telemetry structure.

Table 6-101: All Induct Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_timestampMilliseconds	Timestamp in milliseconds	uint64	8	0*	UINT64_MIN	UINT64_MAX	msec
m_bundleCountEgress	Bundle count egress	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_bundleCountStorage	Bundle count storage	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_bundleByteCountEgress	Bundle byte count egress	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_bundleByteCountStorage	Bundle byte count storage	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_listAllInducts	All inducts list	Induct Telemetry list	variable	*	N/A	N/A	N/A

6.29.2 All Outduct Capabilities Telemetry

Table 6-102: Outduct Capabilities Telemetry defines the Outduct Capabilities Telemetry structure.

Table 6-102: Outduct Capabilities Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
outductCapabilityTelemetryList	Outduct capability telemetry list	Outduct Capability Telemetry list	variable	*	N/A	N/A	N/A

6.29.3 All Outduct Telemetry

Table 6-103: UDP Outduct Telemetry defines the All Outduct Telemetry structure.

Table 6-103: UDP Outduct Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_timestampMilliseconds	Timestamp in milliseconds	uint64	8	0*	UINT64_MIN	UINT64_MAX	msec
m_totalBundlesGivenToOutducts	Total bundles given to outducts	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleBytesGivenToOutducts	Total bundle bytes given to outducts	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalTcpclBundlesReceived	Total TCPCL bundles received	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalTcpclBundleBytesReceived	Total TCPCL bundle bytes received	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalStorageToIngressOpportunisticBundles	Total storage to ingress opportunistic bundles	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalStorageToIngressOpportunisticBundleBytes	Total storage to ingress opportunistic bundle bytes	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalBundlesSuccessfullySent	Total bundles successfully sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleBytesSuccessfullySent	Total bundle bytes successfully sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_listAllOutducts	List all outducts	Outduct Telemetry unique pointer list	variable	*	N/A	N/A	N/A

6.29.4 API Command

Table 6-104: API Command defines the API Command structure.

Table 6-104: API Command

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_apiCall	API call	string	variable	“”*	N/A	N/A	N/A

6.29.5 API Response

Table 6-105: API Response defines the API Response structure.

Table 6-105: API Response

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_success	Success flag	bool	1	FALSE*	FALSE	TRUE	N/A
m_message	Message	string	variable	“”*	N/A	N/A	N/A

6.29.6 BP Over Encapsulation Local Stream Induct Connection Telemetry

Table 6-106: BP Over Encapsulation Local Stream Induct Connection Telemetry defines the BP Over Encapsulation Local Stream Induct Connection Telemetry structure.

Table 6-106: BP Over Encapsulation Local Stream Induct Connection Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
BpOverEncapLocalStreamInductConnection Telemetry : InductConnectionTelemetry	Induct connection telemetry	Induct Connection Telemetry	variable	N/A	N/A	N/A	N/A
m_totalEncapHeaderBytesSent	Total encapsulation header bytes sent	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_totalEncapHeaderBytesReceived	Total encapsulation header bytes received	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_largestEncapHeaderSizeBytesReceived	Largest encapsulation header size bytes received	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_smallestEncapHeaderSizeBytesReceived	Smallest encapsulation header size bytes receive	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_averageEncapHeaderSizeBytesReceived	Average encapsulation header size bytes received	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_totalBundlesSentAndAcked	Total bundles sent and acknowledged	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
m_totalBundleBytesSentAndAcked	Total bundle bytes sent and acknowledged	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_totalBundlesSent	Total bundles sent	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
m_totalBundleBytesSent	Total bundle bytes sent	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_totalBundlesFailedToSend	Total bundles failed to send	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A

6.29.7 BP Over Encapsulation Local Stream Outduct Telemetry

Table 6-107: BP Over Encapsulation Local Stream Outduct Telemetry defines the BP Over Encapsulation Local Stream Outduct Telemetry structure.

Table 6-107: BP Over Encapsulation Local Stream Outduct Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
BpOverEncapLocalStreamOutductTelemetry : OutductTelemetry	Outduct telemetry	Outduct Telemetry	variable	N/A	N/A	N/A	N/A
m_totalEncapHeaderBytesSent	Total encapsulation header bytes sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalEncapHeaderBytesReceived	Total encapsulation header bytes received	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_largestEncapHeaderSizeBytesSent	Largest encapsulation header size bytes sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_smallestEncapHeaderSizeBytesSent	Smallest encapsulation header size bytes sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_averageEncapHeaderSizeBytesSent	Average encapsulation header size bytes sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalBundlesReceived	Total bundles received	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleBytesReceived	Total bundle bytes received	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes

6.29.8 Get Expiring Storage API Command

Table 6-108: Get Expiring Storage API Command defines the Get Expiring Storage API Command structure.

Table 6-108: Get Expiring Storage API Command

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
GetExpiringStorageApiCommand : ApiCommand	API command	API Command	variable	N/A	N/A	N/A	N/A
m_priority	Priority	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_thresholdSecondsFromNow	Threshold seconds from now	uint64	8	0*	UINT64_MIN	UINT64_MAX	sec

6.29.9 Induct Connection Telemetry

Table 6-109: Induct Connection Telemetry defines the Induct Connection Telemetry structure.

Table 6-109: Induct Connection Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_connectionName	Connection name	string	variable	“”*	N/A	N/A	N/A
m_inputName	Input name	string	variable	“”*	N/A	N/A	N/A
m_totalBundlesReceived	Total bundles received	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleBytesReceived	Total bundle bytes received	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes

6.29.10 Induct Telemetry

Table 6-110: Induct Telemetry defines the Induct Telemetry structure.

Table 6-110: Induct Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_convergenceLayer	Convergence layer	string	variable	N/A	N/A	N/A	N/A
m_listInductConnections	Induct connections list	Induct Connection Telemetry unique pointer list	variable	*	N/A	N/A	N/A

6.29.11 LTP Induct Connection Telemetry

Table 6-111: LTP Induct Connection Telemetry defines the LTP Induct Connection Telemetry structure.

Table 6-111: LTP Induct Connection Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
LtpInductConnectionTelemetry : InductConnectionTelemetry	Induct connection telemetry	Induct Connection Telemetry	variable	N/A	N/A	N/A	N/A
m_numReportSegmentTimerExpiredCallbacks	Number of Report Segment Timer Expried Callbacks	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numReportSegmentsUnableToBeIssued	Number of Report Segments Unable to Be Issued	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numReportSegmentsTooLargeAndNeedingSplit	Number of Report Segments Too Large and Needing Split	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numReportSegmentsCreatedViaSplit	Number of report segments created via split	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numGapsFilledByOutOfOrderDataSegments	Number of gaps filled by out of order data segments	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numDelayedFullyClaimedPrimaryReportSegmentsSent	Number of delayed fully claimed primary report segments sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_numDelayedFullyClaimedSecondaryReportSegmentsSent	Number of delayed fully claimed secondary report segments sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numDelayedPartiallyClaimedPrimaryReportSegmentsSent	Number of delayed partially claimed primary report segments sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numDelayedPartiallyClaimedSecondaryReportSegmentsSent	Number of delayed partially claimed secondary report segments sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalCancelSegmentsStarted	Total cancel segments started	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalCancelSegmentSendRetries	Total cancel segment send retries	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalCancelSegmentsFailedToSend	Total cancel segments failed to send	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalCancelSegmentsAcknowledged	Total cancel segments acknowledged	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numRxSessionsCancelledBySender	Number receive sessions cancelled by sender	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numStagnantRxSessionsDeleted	Number of stagnant receive sessions deleted	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_countUdpPacketsSent	UDP packets sent count	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_countRxUdpCircularBufferOverruns	Receive UDP circular buffer overruns count	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_countTxUdpPacketsLimitedByRate	Transmit UDP packets limited by rate count	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.29.12 LTP Outduct Telemetry

Table 6-112: LTP Outduct Telemetry defines the LTP Outduct Telemetry structure.

Table 6-112: LTP Outduct Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
LtpOutductTelemetry : OutductTelemetry	Outduct telemetry	Outduct Telemetry	variable	N/A	N/A	N/A	N/A
m_numCheckpointsExpired	Number of checkpoints expired	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numDiscretionaryCheckpointsNotResent	Number of discretionary checkpoints not resent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numDeletedFullyClaimedPendingReports	Number of deleted fully claimed pending reports	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_totalCancelSegmentsStarted	Total cancel segments started	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalCancelSegmentSendRetries	Total cancel segment send retries	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalCancelSegmentsFailedToSend	Total cancel segments failed	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalCancelSegmentsAcknowledged	Total cancel segments acknowledged	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalPingsStarted	Total pings started	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalPingRetries	Total ping retries	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalPingsFailedToSend	Total pings failed to send	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalPingsAcknowledged	Total pings acknowledged	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numTxSessionsReturnedToStorage	Number of transmit sessions returned to storage	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numTxSessionsCancelledByReceiver	Number of transmit sessions cancelled by receiver	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_countUdpPacketsSent	Count UDP packets sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_countRxUdpCircularBufferOverruns	Count receive UDP circular buffer overruns	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_countTxUdpPacketsLimitedByRate	Count transmit UDP packets limited by rate	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.29.13 Outduct Capability Telemetry

Table 6-113: Outduct Capability Telemetry defines the Outduct Capability Telemetry structure.

Table 6-113: Outduct Capability Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
outductArrayIndex	Outduct array index	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
maxBundlesInPipeline	Max bundles in pipeline	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
maxBundleSizeBytesInPipeline	Max bundle size in bytes in pipeline	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
nextHopNodeId	Next hop node ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
assumedInitiallyDown	Assumed initially down flag	bool	1	FALSE*	FALSE	TRUE	N/A
finalDestinationEidList	Final destination EID list	CBHE EID list	variable	*	N/A	N/A	N/A
finalDestinationNodeIdList	Final destination node ID list	uint64 list	variable	*	N/A	N/A	N/A

6.29.14 Outduct Telemetry

Table 6-114: Outduct Telemetry defines the Outduct Telemetry structure.

Table 6-114: Outduct Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_convergenceLayer	Convergence layer	string	variable	“”*	N/A	N/A	N/A
m_totalBundlesAcked	Total bundles acknowledged	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleBytesAcked	Total bundle bytes acknowledged	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalBundlesSent	Total bundles sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleBytesSent	Total bundle bytes sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalBundlesFailedToSend	Total bundles failed to send	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_linkIsUpPhysically	Link is up physically flag	bool	1	FALSE*	FALSE	TRUE	N/A
m_linkIsUpPerTimeSchedule	Link Is up per time schedule flag	bool	1	FALSE*	FALSE	TRUE	N/A

6.29.15 Ping API Command

Table 6-115: Ping API Command defines the Ping API Command structure.

Table 6-115: Ping API Command

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
PingApiCommand : ApiCommand	API command	API Command	variable	N/A	N/A	N/A	N/A
m_nodeId	Node ID	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_pingServiceNumber	Ping service number	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_bpVersion	BP version	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.29.16 SLIP Over UART Induct Connection Telemetry

Table 6-116: SLIP Over UART Induct Connection Telemetrydefines the LTP Induct Connection Telemetry structure.

Table 6-116: SLIP Over UART Induct Connection Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
SlipOverUartInductConnectionTelemetry : InductConnectionTelemetry	Induct connection telemetry	Induct Connection Telemetry	variable	N/A	N/A	N/A	N/A
m_totalSlipBytesSent	Total SLIP bytes sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_totalSlipBytesReceived	Total SLIP bytes received	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_totalReceivedChunks	Total received chunks	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
m_largestReceivedBytesPerChunk	Largest received bytes per chunk	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_averageReceivedBytesPerChunk	Average received bytes per chunk	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_totalBundlesSentAndAcked	Total bundles sent and acknowledged	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
m_totalBundleBytesSentAndAcked	Total bundle bytes sent and acknowledged	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_totalBundlesSent	Total bundles sent	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
m_totalBundleBytesSent	Total bundle bytes sent	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_totalBundlesFailedToSend	Total bundles failed to send	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A

6.29.17 SLIP Over UART Outduct Telemetry

Table 6-117: SLIP Over UART Outduct Telemetry defines the SLIP Over UART Outduct Telemetry structure.

Table 6-117: SLIP Over UART Outduct Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
SlipOverUartOutductTelemetry : OutductTelemetry	Outduct telemetry	Outduct Telemetry	variable	N/A	N/A	N/A	N/A
m_totalSlipBytesSent	Total SLIP bytes sent	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_totalSlipBytesReceived	Total SLIP bytes received	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_totalReceivedChunks	Total received chunks	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
m_largestReceivedBytesPerChunk	Largest received bytes per chunk	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_averageReceivedBytesPerChunk	Average received bytes per chunk	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes
m_totalBundlesReceived	Total bundles received	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A
m_totalBundleBytesReceived	Total bundle bytes received	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	bytes

6.29.18 Simple TCP (STCP) Induct Connection Telemetry

Table 6-118: STCP Induct Connection Telemetry defines the STCP Induct Connection Telemetry structure.

Table 6-118: STCP Induct Connection Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
StcpInductConnectionTelemetry : InductConnectionTelemetry	Induct connection telemetry	Induct Connection Telemetry	variable	N/A	N/A	N/A	N/A
m_totalStcpBytesReceived	Total STCP bytes received	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes

6.29.19 STCP Outduct Telemetry

Table 6-119: STCP Outduct Telemetry defines the STCP Outduct Telemetry structure.

Table 6-119: STCP Outduct Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
StcpOutductTelemetry : OutductTelemetry	Outduct telemetry	Outduct Telemetry	variable	N/A	N/A	N/A	N/A
m_totalStcpBytesSent	Total STCP bytes sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_numTcpReconnectAttempts	Number of TCP reconnect attempts	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.29.20 Storage Expiring Before Threshold Telemetry

Table 6-120: Storage Expiring Before Threshold Telemetry defines the Storage Expiring Before Threshold Telemetry structure.

Table 6-120: Storage Expiring Before Threshold Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
priority	Priority	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
thresholdSecondsSinceStartOfYear2000	Threshold seconds since start of year 2000	uint64	8	0*	UINT64_MIN	UINT64_MAX	sec

6.29.21 Storage Telemetry

Table 6-121: Storage Telemetry defines the Storage Telemetry structure.

Table 6-121: Storage Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_timestampMilliseconds	Timestamp in milliseconds	uint64	8	0*	UINT64_MIN	UINT64_MAX	msec
m_totalBundlesErasedFromStorageNoCustodyTransfer	Total bundles erased from storage with no custody transfer	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundlesErasedFromStorageWithCustodyTransfer	Total bundles erased from storage with custody transfer	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundlesErasedFromStorageBecauseExpired	Total bundles erased from storage because expired	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundlesRewrittenToStorageFromFailedEgressSend	Total bundles rewritten to storage from failed egress send	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundlesSentToEgressFromStorageReadFromDisk	Total bundles sent to egress from storage read from disk	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleBytesSentToEgressFromStorageReadFromDisk	Total bundle bytes sent to egress from storage read from disk	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalBundlesSentToEgressFromStorageForwardCutThrough	Total bundles sent to egress from storage forward cut-through	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleBytesSentToEgressFromStorageForwardCutThrough	Total bundle bytes sent to egress from storage forward cut-through	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_numRfc5050CustodyTransfers	Number of RFC 5050 custody transfers	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numAcsCustodyTransfers	Number of ACS custody transfers	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numAcsPacketsReceived	Number of ACS packets received	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numBundlesOnDisk	Number of bundles on disk	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_numBundleBytesOnDisk	Number of bundle bytes on disk	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_totalBundleWriteOperationsToDisk	Total bundle write operations to disk	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleByteWriteOperationsToDisk	Total bundle byte write operations to disk	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalBundleEraseOperationsFromDisk	Total bundle erase operations from disk	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleByteEraseOperationsFromDisk	Total bundle byte erased operations from disk	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_usedSpaceBytes	Used space in bytes	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_freeSpaceBytes	Free space in bytes	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes

6.29.22 TCPCL Version 3 (TCPCLv3) Induct Connection Telemetry

Table 6-122: TCPCLv3 Induct Connection Telemetry defines the TCPCLv3 Induct Connection Telemetry structure.

Table 6-122: TCPCLv3 Induct Connection Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
TcpclV3InductConnectionTelemetry : InductConnectionTelemetry	Induct connection telemetry	Induct Connection Telemetry	variable	N/A	N/A	N/A	N/A
m_totalIncomingFragmentsAcked	Total incoming fragments acknowledged	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalOutgoingFragmentsSent	Total outgoing fragments sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundlesSentAndAcked	Total bundles sent and acknowledged	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleBytesSentAndAcked	Total bundle bytes sent and acknowledged	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalBundlesSent	Total bundles sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleBytesSent	Total bundle bytes sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalBundlesFailedToSend	Total bundles failed to send	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.29.23 TCPCLv3 Outduct Telemetry

Table 6-123: TCPCLv3 Outduct Telemetry defines the TCPCLv3 Outduct Telemetry structure.

Table 6-123: TCPCLv3 Outduct Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
TcpclV3OutductTelemetry : OutductTelemetry	Outduct telemetry	Outduct Telemetry	variable	N/A	N/A	N/A	N/A
m_totalFragmentsAcked	Total fragments acknowledged	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalFragmentsSent	Total fragments sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundlesReceived	Total bundles received	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleBytesReceived	Total bundle bytes received	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_numTcpReconnectAttempts	Number of TCP reconnect attempts	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.29.24 TCPCLv4 Induct Connection Telemetry

Table 6-124: TCPCLv4 Induct Connection Telemetry defines the TCPCLv4 Induct Connection Telemetry structure.

Table 6-124: TCPCLv4 Induct Connection Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
TcpclV4InductConnectionTelemetry : InductConnectionTelemetry	Induct connection telemetry	Induct Connection Telemetry	variable	N/A	N/A	N/A	N/A
m_totalIncomingFragmentsAcked	Total incoming fragments acknowledged	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalOutgoingFragmentsSent	Total outgoing fragments sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundlesSentAndAcked	Total bundles sent and acknowledged	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleBytesSentAndAcked	Total bundle bytes sent and acknowledged	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalBundlesSent	Total bundles sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleBytesSent	Total bundle bytes sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalBundlesFailedToSend	Total bundles failed to send	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.29.25 TCPCLv4 Outduct Telemetry

Table 6-125: TCPCLv4 Outduct Telemetry defines the TCPCLv4 Outduct Telemetry structure.

Table 6-125: TCPCLv4 Outduct Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
TcpclV4OutductTelemetry : OutductTelemetry	Outduct telemetry	Outduct Telemetry	variable	N/A	N/A	N/A	N/A
m_totalFragmentsAcked	Total fragments acknowledged	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalFragmentsSent	Total fragments sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundlesReceived	Total bundles received	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalBundleBytesReceived	Total bundle bytes received	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_numTcpReconnectAttempts	Number of TCP reconnect attempts	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.29.26 User Datagram Protocol (UDP) Induct Connection Telemetry

Table 6-126: UDP Induct Connection Telemetry defines the UDP Induct Connection Telemetry structure.

Table 6-126: UDP Induct Connection Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
UdpInductConnectionTelemetry : InductConnectionTelemetry	Induct connection telemetry	Induct Connection Telemetry	variable	N/A	N/A	N/A	N/A
m_countCircularBufferOverruns	Circular buffer overruns count	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.29.27 UDP Outduct Telemetry

Table 6-127: UDP Outduct Telemetry defines the UDP Outduct Telemetry structure.

Table 6-127: UDP Outduct Telemetry

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
UdpOutductTelemetry : OutductTelemetry	Outduct telemetry	Outduct Telemetry	variable	N/A	N/A	N/A	N/A
m_totalPacketsSent	Total packets sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalPacketBytesSent	Total packet bytes sent	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalPacketsDequeuedForSend	Total packets dequeued for send	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A
m_totalPacketBytesDequeuedForSend	Total packet bytes dequeued for send	uint64	8	0*	UINT64_MIN	UINT64_MAX	bytes
m_totalPacketsLimitedByRate	Total packets limited by rate	uint64	8	0*	UINT64_MIN	UINT64_MAX	N/A

6.29.28 Update BPSec API Command

Table 6-128: Update BPSec API Command defines the Update BPSec API Command structure.

Table 6-128: Update BPSec API Command

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
UpdateBpSecApiCommand : ApiCommand	API command	API Command	variable	N/A	N/A	N/A	N/A
m_bpSecJson	BPSec JSON	string	variable	“”*	N/A	N/A	N/A

6.29.29 Upload Contact Plan API Command

Table 6-129: Upload Contact Plan API Command defines the Upload Contact Plan API Command structure.

Table 6-129: Upload Contact Plan API Command

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
UploadContactPlanApiCommand : ApiCommand	API command	API Command	variable	N/A	N/A	N/A	N/A
m_contactPlanJson	Contact plan JSON	string	variable	“”*	N/A	N/A	N/A

6.29.30 Set Link Down API Command

Table 6-130: Set Link Down API Command defines the Set Link Down API Command structure.

Table 6-130: Set Link Down API Command

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
SetLinkDownApiCommand : ApiCommand	API command	API Command	variable	N/A	N/A	N/A	N/A
m_index	Index	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A

6.29.31 Set Link Up API Command

Table 6-131: Set Link Up API Command defines the Set Link Up API Command structure.

Table 6-131: Set Link Up API Command

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
SetLinkUpApiCommand : ApiCommand	API command	API Command	variable	N/A	N/A	N/A	N/A
m_index	Index	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A

6.29.32 Set Max Send Rate API Command

Table 6-132: Set Max Send Rate API Command defines the Set Max Send Rate API Command structure.

Table 6-132: Set Max Send Rate API Command

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
SetMaxSendRateApiCommand : ApiCommand	API command	API Command	variable	N/A	N/A	N/A	N/A
m_rateBitsPerSec	Rate bits per second	unit64	8	0*	UINT64_ MIN	UINT64_ MAX	bits / sec
m_outduct	Outduct	uint64	8	0*	UINT64_ MIN	UINT64_ MAX	N/A

6.30 Transmission Control Protocol Asynchronous Sender Structures

The following section defines data structures specific to TCP Asynchronous Sender implementation.

6.30.1 TCP Asynchronous Sender Element

Table 6-133: TCP Asynchronous Sender Element defines the TCP Asynchronous Sender Element structure.

Table 6-133: TCP Asynchronous Sender Element

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_userData	User data	uint8 vector	variable	*	N/A	N/A	N/A
m_constBufferVec	Constant buffer vector	Boost ASIO constant buffer vector	variable	*	N/A	N/A	N/A
m_underlyingDataVecHeaders	Underlying data vector headers	uint8 vector vector	variable	*	N/A	N/A	N/A
m_underlyingDataVecBundle	Underlying data vector bundle	uint8 vector	variable	*	N/A	N/A	N/A
m_underlyingDataZmqBundle	Underlying data ZeroMQ bundle	ZMQ Message unique pointer	N/A	*	N/A	N/A	N/A
m_onSuccessfulSendCallback ByIoServiceThreadPtr	On successful send callback by I/O service thread pointer	On Successful Send Callback by I/O Service Thread pointer	N/A	*	N/A	N/A	N/A

6.31 Transmission Control Protocol Convergence Layer Data Structures

The following section defines data structures specific to TCPCL implementation.

6.31.1 TCPCL Contact Header Flags

Table 6-134: TCPCL Contact Header Flags defines the TCPCL Contact Header Flags bit-packed 32-bit structure.

Table 6-134: TCPCL Contact Header Flags

Mnemonic	Bit Position	Description
REQUEST_ACK_OF_BUNDLE_SEGMENTS	0	Request acknowledgment of bundle segments
REQUEST_ENABLING_OF_REACTIVE_FRAGMENTATION	1	Request enabling of reactive fragmentation
SUPPORT_BUNDLE_REFUSAL	2	Indicate support for bundle refusal
REQUEST_SENDING_OF_LENGTH_MESSAGES	3	Request sending of LENGTH messages
Spare	4-31	Unassigned

6.32 Universal Asynchronous Receiver/Transmitter Interface Data Structures

The following section defines data structures specific to Universal Asynchronous Receiver/Transmitter (UART) Interface implementation.

6.32.1 Serial Send Element

Table 6-135: Serial Send Element defines the Serial Send Element structure.

Table 6-135: Serial Send Element

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
m_userData	User Data	uint8 vector	variable	*	N/A	N/A	N/A
m_slipEncodedBundle	SLIP encoded bundle	Common Uint8 Padded Vector	variable	*	N/A	N/A	N/A
m_underlyingDataVec Bundle	Underlying data vector bundle	Common Uint8 Padded Vector	variable	*	N/A	N/A	N/A
m_underlyingDataZmq Bundle	Underlying data ZeroMQ bundle	ZMQ Message	N/A	*	N/A	N/A	N/A

6.33 ZeroMQ Structures

The following section defines data structures specific to ZeroMQ implementation.

6.33.1 Receive Buffer Size

Definition of the Receive Buffer Size structure is dependent on definition of the ZMQ_CPP11 symbolic constant.

Table 6-136: Receive Buffer Size (ZMQ_CPP11 Defined) defines the Receive Buffer Size structure when ZMQ_CPP11 is defined.

Table 6-136: Receive Buffer Size (ZMQ_CPP11 Defined)

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
size	Size	size	8	*	N/A	N/A	bytes
untruncated_size	Untruncated size	size	8	*	N/A	N/A	bytes

6.33.2 ZeroMQ Event

Definition of the ZeroMQ Event structure is dependent on definition of the ZMQ_VERSION symbolic constant.

Table 6-137: ZeroMQ Event defines the ZeroMQ Event structure when ZMQ_VERSION ≥ ZMQ_MAKE_VERSION(4, 1, 0).

Table 6-137: ZeroMQ Event

Mnemonic	Description	Type	Size (bytes)	Init	Min	Max	Units
event	Event	uint16	2	0*	UINT16_MIN	UINT16_MAX	N/A
value	Value	int32	4	0*	INT32_MIN	INT32_MAX	N/A

7.0 Input/Output Data

This section defines input and output (I/O) formats and units of measure.

7.1 External I/O

This section describes the structure and contents of node-to-node transmissions.

Any data transmitted from one HDTN node to another can be received by that node from another.

7.1.1 LTP Segments

NOTE: LTP segments make use of Self-Delimiting Numeric Values (SDNVs). More information on SDNVs can be found in Internet Research Task Force (IRTF) Request for Comments (RFC) 6256 Using Self-Delimiting Numeric Values in Protocols.

LTP is a convergence layer protocol for DTN that provides retransmission-based reliability, operating directly over a link-layer protocol or over UDP at the transport layer.

LTP Segments have the following formats:

- LTP Data Segment
- LTP Report Segment
- LTP Report Acknowledgement Segment
- LTP Cancel Segment
- LTP Cancel Acknowledgement Segment

LTP Segments are composed of a header, header extensions (optional), segment contents (optional), and trailer extensions (optional). Table 7-1: LTP Segment defines the structure of the LTP Segment.

Table 7-1: LTP Segment

Bit	0	1	2	3	4	5	6	7
Octet 0	version number				segment type flags			
Octet 1	session ID (byte 0)							
…	…							
Octet 16	session ID (byte 15)							
Octet 17	hdr. Ext. count				trlr. ext. count			
Octet 18	header extensions (byte 0)							
…	…							
Octet 18+X	header extensions (byte X)							
Octet 19+X	segment content (byte 0)							
…	…							
Octet 19+X+Y	segment content (byte Y)							
Octet 20+X+Y	trailer extensions (byte 0)							
…	…							
Octet 20+X+Y+Z	trailer extensions (byte Z)							

Version Number is a 4-bit field indicating the LTP version in use. This field is always set to 0b0000.

Segment Type Flags is a bit-packed 4-bit structure. Table 7-2: LTP Segment – Segment Type Flags defines the structure of Segment Type Flags.

Table 7-2: LTP Segment – Segment Type Flags

Mnemonic	Bit Position	Description
CTRL	0 (MSB)	Control flag
EXC	1	Exception flag
Flag 1	2	Flag 1
Flag 0	3 (LSB)	Flag 0

The values indicated by these flags are captured in 4.13.9 LTP Segment Type Flags.

Session ID is a field of variable length that uniquely identifies the session to which the data segment belongs. Session ID is the concatenation of two SDNVs. Table 7-3: LTP Segment – Session ID defines the structure of Session ID.

Table 7-3: LTP Segment – Session ID

Bit	0	1	2	3	4	5	6	7
Octet 0	session originator (byte 0)							
…	…							
Octet V	session originator (byte V)							
Octet 1+V	session number (byte 0)							
…	…							
Octet 1+V+W	session number (byte W)							

Session Originator indicates the engine ID of the sender.

Session Number is generated by the sender to uniquely identify the session when the session is initiated.

Header Extension Count is a 4-bit field indicating the number of extensions preceding the segment content in the data segment. HDTN implementation of LTP does not utilize header extensions. This field is always set to a value of 0.

Trailer Extension Count is a 4-bit field indicating the number of extensions following the segment content in the data segment. HDTN implementation of LTP does not utilize trailer extensions. This field is always set to a value of 0.

7.1.1.1 LTP Data Segment

For LTP Data Segments, segment content is composed of a client service ID, offset, length, checkpoint serial number, report serial number, and client service data. Table 7-4: LTP Data Segment – Segment Content defines the structure of Segment Content.

Table 7-4: LTP Data Segment – Segment Content

Bit	0	1	2	3	4	5	6	7
Octet 0	client service ID (byte 0)							
…	…							
Octet U	client service ID (byte U)							
Octet 1+U	offset (byte 0)							
…	…							
Octet 1+U+V	offset (byte V)							
Octet 2+U+V	length (byte 0)							
…	…							
Octet 2+U+V+W	length (byte W)							
Octet 3+U+V+W	checkpoint serial number (byte 0)							
…	…							
Octet 3+U+V+W+X	checkpoint serial number (byte X)							
Octet 4+U+V+W+X	report serial number (byte 0)							
…	…							
Octet 4+U+V+W+X+Y	report serial number (byte Y)							
Octet 5+U+V+W+X+Y	client service data (byte 0)							
…	…							
Octet 5+U+V+W+X+Y+Z	client service data (byte Z)							

Client Service ID is a SDNV indicating the segment destination service.

Offset is a SDNV indicating the position of the segment data within the block being transmitted.

Length is a SDNV indicating the length of the Client Service Data in bytes.

Checkpoint Serial Number is a SDNV that uniquely identifies the checkpoint within the session.

Report Serial Number is a SDNV identifying the checkpoint that caused the data segment to be queued for transmission.

Client Service Data is an array of bytes copied from the original client service data block beginning at the position indicated by Offset.

7.1.1.2 LTP Report Segment

For LTP Report Segments, segment content is composed of a report serial number, checkpoint serial number, upper bound, lower bound, reception claim count, and reception claims. Table 7-5: LTP Report Segment – Segment Content defines the structure of Segment Content.

Table 7-5: LTP Report Segment – Segment Content

Bit	0	1	2	3	4	5	6	7
Octet 0	report serial number (byte 0)							
…	…							
Octet U	report serial number (byte U)							
Octet 1+U	checkpoint serial number (byte 0)							
…	…							
Octet 1+U+V	checkpoint serial number (byte V)							
Octet 2+U+V	upper bound (byte 0)							
…	…							
Octet 2+U+V+W	upper bound (byte W)							
Octet 3+U+V+W	lower bound (byte 0)							
…	…							
Octet 3+U+V+W+X	lower bound (byte X)							
Octet 4+U+V+W+X	reception claim count (byte 0)							
…	…							
Octet 4+U+V+W+X+Y	reception claim count (byte Y)							
Octet 5+U+V+W+X+Y	reception claims (byte 0)							
…	…							
Octet 5+U+V+W+X+Y+Z	reception claims (byte Z)							

Report Serial Number is a SDNV that uniquely identifies the report among all reports issued by a receiver during a session.

Checkpoint Serial Number is a SDNV that uniquely identifies the checkpoint that caused a report to be issued.

Upper Bound is a SDNV indicating the size of the block prefix related to the segment reception claims.

Lower Bound is a SDNV indicating the size of the block prefix unrelated to the segment reception claims.

Reception Claim Count is a SDNV indicating the number of Reception Claims in the report segment.

Receptions Claims are a series of offset and length value pairs. Table 7-6: LTP Report Segment – Reception Claim defines the structure of Reception Claim.

Table 7-6: LTP Report Segment – Reception Claim

Bit	0	1	2	3	4	5	6	7
Octet 0	offset (byte 0)							
…	…							
Octet S	offset (byte S)							
Octet 1+S	length (byte 0)							
…	…							
Octet 1+S+T	length (byte T)							

Offset is a SDNV indicating the byte position of successfully received data relative to the Lower Bound of the report.

Length is a SDNV indicating the number of contiguous bytes of data successfully received.

7.1.1.3 LTP Report Acknowledgement Segment

For LTP Report Acknowledgement Segments, segment content is composed of a report serial number. Table 7-7: LTP Report Acknowledgement Segment – Segment Content defines the structure of Segment Content.

Table 7-7: LTP Report Acknowledgement Segment – Segment Content

Bit	0	1	2	3	4	5	6	7
Octet 0	report serial number (byte 0)							
…	…							
Octet X	report serial number (byte X)							

Report Serial Number is a SDNV reflecting the serial number in the acknowledged report segment.

7.1.1.4 LTP Cancel Segment

For LTP Cancel Segments, segment content is composed of a reason code. Table 7-8: LTP Cancel Segment – Segment Content defines the structure of Segment Content.

Table 7-8: LTP Cancel Segment – Segment Content

Bit	0	1	2	3	4	5	6	7
Octet 0	reason code							

Reason Code is an 8-bit integer value indicating the reason the session is being canceled. Table 7-9: LTP Cancel Segment – Reason Codes defines Reason Code enumerated 8-bit integer values.

Table 7-9: LTP Cancel Segment – Reason Codes

Mnemonic	Value	Description
USER_CANCELLED	0x00	Client service canceled session
UNREACHABLE	0x01	Unreachable client service
RLEXC	0x02	Retransmission limit exceeded
MISCOLORED	0x03	Received either: • a red-part data segment at block offset above any green-part data segment offset • a green-part data segment at block offset below any red-part data segment offset
SYSTEM_CANCELLED	0x04	A system error condition caused unexpected session termination
RXMTCYCEXC	0x05	Exceeded the retransmission-cycles limit
RESERVED	0x06-0xFF	Reserved

7.1.1.5 LTP Cancel Acknowledgement Segment

LTP Cancel Acknowledgement Segments have no segment content data.

7.1.2 STCP

Simple TCP is a convergence layer protocol for DTN that bridges the gap between the DTN Bundle Protocol at the application layer and TCP at the transport layer.

STCP messages have the following formats:

- STCP Bundle
- STCP Keepalive

7.1.2.1 STCP Bundle

The STCP Bundle is composed of bundle length and bundle data. Table 7-10: STCP Bundle defines the structure of STCP Bundle.

Table 7-10: STCP Bundle

Bit	0	1	2	3	4	5	6	7
Octet 0	bundle length (byte 0)							
…	…							
Octet 3	bundle length (byte 3)							
Octet 4	bundle data (byte 0)							
…	…							
Octet 4+X	bundle data (byte X)							

Bundle Length is a uint32 value indicating the length of the bundle data in bytes.

Bundle Data is a payload of variable length (as indicated by Bundle Length).

7.1.2.2 STCP Keepalive

The STCP Keepalive is composed of bundle length. Table 7-11: STCP Keepalive defines the structure of STCP Keepalive.

Table 7-11: STCP Keepalive

Bit	0	1	2	3	4	5	6	7
Octet 0	bundle length (byte 0)							
…	…							
Octet 3	bundle length (byte 3)							

Bundle Length is a uint32 value set to 0 for STCP Keepalive messages.

7.1.3 TCPCLv3

TCP Convergence Layer version 3 bridges the gap between the DTN Bundle Protocol at the application layer and TCP at the transport layer.

TCPCLv3 messages have the following formats:

- TCPCLv3 Contact Header
- TCPCLv3 Bundle Data Transmission
- TCPCLv3 Bundle Acknowledgement
- TCPCLv3 Bundle Refusal
- TCPCLv3 Bundle Length
- TCPCLv3 Keepalive
- TCPCLv3 Shutdown Message

7.1.3.1 TCPCLv3 Contact Header

The TCPCLv3 Contact Header is composed of a magic number, version, binary flags, keepalive interval, local EID length, and local EID. Table 7-12: TCPCLv3 Contact Header defines the structure of TCPCLv3 Contact Header.

Table 7-12: TCPCLv3 Contact Header

Bit	0	1	2	3	4	5	6	7
Octet 0	magic (byte 0)							
Octet 1	magic (byte 1)							
Octet 2	magic (byte 2)							
Octet 3	magic (byte 3)							
Octet 4	version							
Octet 5	flags							
Octet 6	keepalive interval (byte 0)							
Octet 7	keepalive interval (byte 1)							
Octet 8	local EID length (byte 0)							

Bit	0	1	2	3	4	5	6	7
…	…							
Octet 8+X	local EID length (byte X)							
Octet 9+X	local EID (byte 0)							
…	…							
Octet 9+X+Y	local EID (byte Y)							

Magic is a four-byte field containing a constant value of 0x6474 6E21 (ASCII text “dtn!”)

Version is an eight-bit unsigned integer containing a constant value of 4.

Flags is a bit-packed 8-bit structure. Table 7-13: TCPCLv3 Contact Header – Flags defines the structure of Flags.

Table 7-13: TCPCLv3 Contact Header – Flags

Bit Position	Description
0 (MSB)	Reserved for future use
1	Reserved for future use
2	Reserved for future use
3	Reserved for future use
4	Length message transmission is requested
5	Bundle refusal is supported
6	Enabling of reactive fragmentation is requested
7 (LSB)	Acknowledgement of bundle segments is requested

Keepalive Interval is a 16-bit integer value indicating the number of seconds between exchanges of KEEPALIVE messages during the session.

Local EID Length is a SDNV indicating the length of Local EID in bytes.

Local EID is a value of variable length (as indicated by Local EID Length) indicating the endpoint of the sending node in `<scheme name>:<scheme-specific part>` format.

7.1.3.2 TCPCLv3 Bundle Data Transmission

The TCPCLv3 Bundle Data Transmission is composed of a message type, message flags, length, and contents. Table 7-14: TCPCLv3 Bundle Data Transmission defines the structure of TCPCLv3 Bundle Data Transmission.

Table 7-14: TCPCLv3 Bundle Data Transmission

Bit	0	1	2	3	4	5	6	7
Octet 0	type				flags			
Octet 1	length (byte 0)							
…	…							
Octet 1+X	length (byte X)							
Octet 2+X	contents (byte 0)							
…	…							
Octet 2+X+Y	contents (byte Y)							

Type is a 4-bit integer value indicating the type of message that follows. Type values are defined in Table 4-55: TCPCL Message Type Byte Codes. For Data Transmission, this field is set to DATA_SEGMENT.

Flags is a bit-packed 4-bit structure. Table 7-15: TCPCLv3 Bundle Data Transmission –Flags defines the structure of Flags.

Table 7-15: TCPCLv3 Bundle Data Transmission –Flags

Mnemonic	Bit Position	Description
RESERVED	0 (MSB)	Reserved for future use.
RESERVED	1	Reserved for future use.
S	2	This is the first segment of a new bundle.
E	3 (LSB)	This is the last segment of a bundle.

Length is a SDNV indicating the length of Contents in bytes.

Contents is a payload of variable length (as indicated by Length.)

7.1.3.3 TCPCLv3 Bundle Acknowledgement

The TCPCLv3 Bundle Acknowledgement is composed of a message type, message flags, and acknowledged length. Table 7-16: TCPCLv3 Bundle Acknowledgement defines the structure of TCPCLv3 Bundle Acknowledgement.

Table 7-16: TCPCLv3 Bundle Acknowledgement

Bit	0	1	2	3	4	5	6	7
Octet 0	type				flags			
Octet 1	acknowledged length (byte 0)							
…	…							
Octet 1+X	acknowledged length (byte X)							

Type is a 4-bit integer value indicating the type of message that follows. Type values are defined in Table 4-55: TCPCL Message Type Byte Codes. For Bundle Acknowledgement, this field is set to ACK_SEGMENT.

Flags is a bit-packed 4-bit structure that is always set to 0.

Acknowledged Length is a SDNV indicating the cumulative length of the current bundle segments received in bytes.

7.1.3.4 TCPCLv3 Bundle Refusal

The TCPCLv3 Bundle Refusal is composed of a message type and reason code. Table 7-17: TCPCLv3 Bundle Refusal defines the structure of TCPCLv3 Bundle Refusal.

Table 7-17: TCPCLv3 Bundle Refusal

Bit	0	1	2	3	4	5	6	7
Octet 0	type				rcode			

Type is a 4-bit integer value indicating the type of message that follows. Type values are defined in Table 4-55: TCPCL Message Type Byte Codes. For Bundle Refusal, this field is set to REFUSE_BUNDLE.

Rcode is a 4-bit unsigned integer value indicating the reason the bundle is being refused. Table 7-18: TCPCLv3 Bundle Refusal – Reason Codes defines Reason Code enumerated 4-bit integer values.

Table 7-18: TCPCLv3 Bundle Refusal – Reason Codes

Value	**Description**
0x0	Reason unknown or unspecified
0x1	Receiver has the complete bundle
0x2	Receiver resources are exhausted
0x3	Receiver encountered a problem and requires retransmission of the bundle in its entirety
0x4-0x7	Unassigned
0x8-0xF	Reserved for future use

7.1.3.5 TCPCLv3 Bundle Length

The TCPCLv3 Bundle Length is composed of a message type, message flags, and total bundle length. Table 7-19: TCPCLv3 Bundle Length defines the structure of TCPCLv3 Bundle Length.

Table 7-19: TCPCLv3 Bundle Length

Bit	0	1	2	3	4	5	6	7
Octet 0	type				flags			
Octet 1	total bundle length (byte 0)							
…	…							
Octet 1+X	total bundle length (byte X)							

Type is a 4-bit integer value indicating the type of message that follows. Type values are defined in Table 4-55: TCPCL Message Type Byte Codes. For Bundle Length, this field is set to LENGTH.

Flags is a bit-packed 4-bit structure that is always set to 0.

Total Bundle Length is a SDNV indicating the cumulative length of the next bundle to be sent in bytes.

7.1.3.6 TCPCLv3 Keepalive

The TCPCLv3 Keepalive is composed of a message type with no additional data. Table 7-20: TCPCLv3 Keepalive defines the structure of TCPCLv3 Keepalive.

Table 7-20: TCPCLv3 Keepalive

Bit	0	1	2	3	4	5	6	7
Octet 0	type				0			

Type is a 4-bit integer value indicating the type of message that follows. Type values are defined in Table 4-55: TCPCL Message Type Byte Codes. For Bundle Length, this field is set to KEEPALIVE.

7.1.3.7 TCPCLv3 Shutdown Message

The TCPCLv3 Shutdown is composed of a message type, message flags, reason, and reconnection delay. Table 7-21: TCPCLv3 Shutdown Message defines the structure of TCPCLv3 Shutdown Message.

Table 7-21: TCPCLv3 Shutdown Message

Bit	0	1	2	3	4	5	6	7
Octet 0	type				flags			
Octet 1	reason							
Octet 2	reconnection delay (byte 0)							
…	…							
Octet 2+X	reconnection delay (byte X)							

Type is a 4-bit integer value indicating the type of message that follows. Type values are defined in Table 4-55: TCPCL Message Type Byte Codes. For Bundle Length, this field is set to SHUTDOWN.

Flags is a bit-packed 4-bit structure. Table 7-22: TCPCLv3 Bundle Data Transmission –Flags defines Flags enumerated 4-bit integer values.

Table 7-22: TCPCLv3 Bundle Data Transmission –Flags

Mnemonic	**Bit Position**	**Description**
RESERVED	0 (MSB)	Reserved for future use
RESERVED	1	Reserved for future use
R	2	This message contains a reason code
D	3 (LSB)	This message contains a reconnection delay

Reason is an optional 8-bit unsigned integer indicating the reason for terminating the session. Reason values are defined in Table 4-56: TCPCL Shutdown Reason Codes.

Reconnection Delay is an optional SDNV indicating the amount of time the receiving node must wait before attempting to re-establish the connection.

7.1.4 TCPCLv4

TCP Convergence Layer version 4 bridges the gap between the DTN Bundle Protocol at the application layer and TCP at the transport layer.

TCPCLv4 messages have the following formats:

- TCPCLv4 Contact Header
- TCPCLv4 Extension Items
- TCPCLv4 Session Initialization Message
- TCPCLv4 Data Transmission
- TCPCLv4 Data Acknowledgement
- TCPCLv4 Transfer Refusal
- TCPCLv4 Message Rejection
- TCPCLv4 Session Upkeep
- TCPCLv4 Session Termination Message

7.1.4.1 TCPCLv4 Contact Header

The TCPCLv4 Contact Header is composed of a magic number, version, and binary flags. Table 7-23: TCPCLv4 Contact Header defines the structure of TCPCLv4 Contact Header.

Table 7-23: TCPCLv4 Contact Header

Bit	0	1	2	3	4	5	6	7
Octet 0	magic (byte 0)							
Octet 1	magic (byte 1)							
Octet 2	magic (byte 2)							
Octet 3	magic (byte 3)							
Octet 4	version							
Octet 5	flags							

Magic is a four-byte field containing a constant value of 0x6474 6E21 (ASCII text “dtn!”)

Version is an eight-bit unsigned integer containing a constant value of 4.

Flags is a bit-packed 8-bit structure. Table 7-24: TCPCLv4 Contact Header – Flags defines Flags enumerated 8-bit integer values.

Table 7-24: TCPCLv4 Contact Header – Flags

Mnemonic	Bit Position	Description
RESERVED	0 (MSB)	Reserved for future use
RESERVED	1	Reserved for future use
RESERVED	2	Reserved for future use
RESERVED	3	Reserved for future use
RESERVED	4	Reserved for future use
RESERVED	5	Reserved for future use
RESERVED	6	Reserved for future use
CAN_TLS	7 (LSB)	The sending peer has enabled TLS security

7.1.4.2 TCPCLv4 Extension Items

TCPCLv4 Extension Items is composed of item flags, item type, item length, and item value. Table 7-25: TCPCLv4 Extension Items defines the structure of TCPCLv4 Extension Items.

Table 7-25: TCPCLv4 Extension Items

Bit	0	1	2	3	4	5	6	7
Octet 0	item flags							
Octet 1	item type (byte 0)							
Octet 2	item type (byte 1)							
Octet 3	item length (byte 0)							
Octet 4	item length (byte 1)							
Octet 5	item value (byte 0)							
…	…							
Octet 5+Y	item value (byte Y)							

Item Flags is a bit-packed 8-bit structure. Table 7-26: TCPCLv4 Extension Items – Item Flags defines Item Flags enumerated 8-bit integer values.

Table 7-26: TCPCLv4 Extension Items – Item Flags

Mnemonic	Bit Position	Description
RESERVED	0 (MSB)	Reserved for future use
RESERVED	1	Reserved for future use
RESERVED	2	Reserved for future use
RESERVED	3	Reserved for future use
RESERVED	4	Reserved for future use
RESERVED	5	Reserved for future use
RESERVED	6	Reserved for future use
CRITICAL	7 (LSB)	Receiving peer must handle the extension item

Item Type is a 16-bit unsigned integer value indicating the type of the extension item. Table 7-27: TCPCLv4 Extension Items – Transfer Extension Type Codes defines Transfer Extension Type Codes enumerated 16-bit integer values.

Table 7-27: TCPCLv4 Extension Items – Transfer Extension Type Codes

Value	Description
0x0000	Reserved
0x0001	Transfer length extension
0x0002-0x7FFF	Unassigned
0x8000-0xFFFF	Reserved for private or experimental use

Item Length is a 16-bit unsigned integer value indicating the length of Item Value in bytes.

Item Value is a field of variable length (as indicated by Item Length) interpreted depending on the value of Item Type.

7.1.4.3 TCPCLv4 Session Initialization Message

The TCPCLv4 Session Initialization message is composed of a header, keepalive interval, segment MRU, transfer MRU, node ID length, node ID data, session extension items length, and session extension items. Table 7-28: TCPCLv4 Session Initialization Message defines the structure of TCPCLv4 Session Initialization Message.

Table 7-28: TCPCLv4 Session Initialization Message

Bit	0	1	2	3	4	5	6	7
Octet 0	message header							
Octet 1	keepalive interval (byte 0)							
Octet 2	keepalive interval (byte 1)							
Octet 3	segment MRU (byte 0)							
…	…							
Octet 10	segment MRU (byte 7)							
Octet 11	transfer MRU (byte 0)							
…	…							
Octet 18	transfer MRU (byte 7)							
Octet 19	node ID length (byte 0)							
Octet 20	node ID length (byte 1)							
Octet 21	node ID data (byte 0)							
	…							
Octet 21+X	node ID data (byte X)							
Octet 22+X	session extension items length (byte 0)							
…	…							

Bit	0	1	2	3	4	5	6	7
Octet 25+X	session extension items length (byte 3)							
Octet 26+X	session extension items (byte 0)							
…	…							
Octet 26+X+Y	session extension items (byte Y)							

Message Header is an 8-bit integer value indicating the type of message that follows. Message Header values are defined in Table 4-63: TCPCLv4 Message Type Byte Codes. For Session Initialization messages, this field is set to SESS_INIT.

Keepalive Interval is a 16-bit unsigned integer value indicating the minimum interval (in seconds) for periodic transmission of Session Upkeep messages proposed by the sender.

Segment MRU is a 64-bit unsigned integer value indicating the largest allowable single-segment data payload size to be received during this session.

Transfer MRU is a 64-bit unsigned integer value indicating the largest allowable total-bundle data size to be received during this session.

Node ID Length is a 16-bit unsigned integer value indicating the length in bytes of Node ID Data.

Node ID Data is a UTF-8 encoded value of variable length (as indicated by Node ID Length) indicating the node ID of the entity that sent the Session Initialization message.

Session Extension Items Length is a 16-bit unsigned integer indicating the length in bytes of Session Extension Items.

Session Extension Items is a series of TCPCLv4 Extension Items as documented in 7.1.4.2 TCPCLv4 Extension Items.

7.1.4.4 TCPCLv4 Data Transmission

The TCPCLv4 Data Transmission is composed of a header, message flags, transfer ID, transfer extension items length, transfer extension items, data length, and data contents. Table 7-29: TCPCLv4 Data Transmission defines the structure of TCPCLv4 Data Transmission.

Table 7-29: TCPCLv4 Data Transmission

Bit	0	1	2	3	4	5	6	7
Octet 0	message header							
Octet 1	message flags							
Octet 2	transfer ID (byte 0)							
…	…							
Octet 9	transfer ID (byte 7)							
Octet 10	transfer extension items length (byte 0)							
…	…							
Octet 13	transfer extension items length (byte 3)							
Octet 14	transfer extension items (byte 0)							
…	…							
Octet 14+X	transfer extension items (byte X)							
Octet 15+X	data length (byte 0)							
…	…							
Octet 22+X	data length (byte 7)							
Octet 23+X	data contents (byte 0)							
…	…							
Octet 23+X+Y	data contents (byte Y)							

Message Header is an 8-bit integer value indicating the type of message that follows. Message Header values are defined in Table 4-63: TCPCLv4 Message Type Byte Codes. For Data Transmission, this field is set to XFER_SEGMENT.

Message Flags is a bit-packed 8-bit structure. Table 7-30: TCPCLv4 Data Transmission – Message Flags defines the structure of Message Flags.

Table 7-30: TCPCLv4 Data Transmission – Message Flags

Mnemonic	Bit Position	Description
RESERVED	0 (MSB)	Reserved for future use
RESERVED	1	Reserved for future use
RESERVED	2	Reserved for future use
RESERVED	3	Reserved for future use
RESERVED	4	Reserved for future use
RESERVED	5	Reserved for future use
START	6	This is the first segment of the transfer
END	7 (LSB)	This is the last segment of the transfer

Transfer ID is a 64-bit unsigned integer value identifying the transfer being made.

Transfer Extension Items Length is a 32-bit unsigned integer value indicating the length of Transfer Extension Items in bytes.

Transfer Extension Items is a series of TCPCLv4 Extension Items as documented in 7.1.4.2 TCPCLv4 Extension Items.

Data Length is a 64-bit unsigned integer indicating the size of Data Contents in bytes.
Data Contents is a payload of variable length (as indicated by Data Length.)

7.1.4.5 TCPCLv4 Data Acknowledgement

The TCPCLv4 Data Acknowledgement is composed of a header, message flags, transfer ID, and acknowledged length. Table 7-31: TCPCLv4 Data Acknowledgement defines the structure of TCPCLv4 Data Acknowledgement.

Table 7-31: TCPCLv4 Data Acknowledgement

Bit	0	1	2	3	4	5	6	7
Octet 0	message header							
Octet 1	message flags							
Octet 2	transfer ID (byte 0)							
…	…							
Octet 9	transfer ID (byte 7)							
Octet 10	acknowledged length (byte 0)							
…	…							
Octet 17	acknowledged length (byte 7)							

Message Header is an 8-bit integer value indicating the type of message that follows. Message Header values are defined in Table 4-63: TCPCLv4 Message Type Byte Codes. For Data Acknowledgement, this field is set to XFER_ACK.

Message Flags is a bit-packed 8-bit structure. Table 7-32: TCPCLv4 Data Acknowledgement – Message Flags defines the structure of Message Flags.

Table 7-32: TCPCLv4 Data Acknowledgement – Message Flags

Mnemonic	**Bit Position**	**Description**
RESERVED	0 (MSB)	Reserved for future use
RESERVED	1	Reserved for future use
RESERVED	2	Reserved for future use
RESERVED	3	Reserved for future use
RESERVED	4	Reserved for future use
RESERVED	5	Reserved for future use
START	6	This is the first segment of the transfer
END	7 (LSB)	This is the last segment of the transfer

Transfer ID is a 64-bit unsigned integer value identifying the transfer being acknowledged.

Acknowledged Length is a 64-bit unsigned integer value indicating the total number of bytes in the transfer being acknowledged.

7.1.4.6 TCPCLv4 Transfer Refusal

The TCPCLv4 Transfer Refusal is composed of a header, reason code, and transfer ID. Table 7-33: TCPCLv4 Data Acknowledgement defines the structure of TCPCLv4 Data Acknowledgement.

Table 7-33: TCPCLv4 Data Acknowledgement

Bit	0	1	2	3	4	5	6	7
Octet 0	message header							
Octet 1	reason code							
Octet 2	transfer ID (byte 0)							
…	…							
Octet 9	transfer ID (byte 7)							

Message Header is an 8-bit integer value indicating the type of message that follows. Message Header values are defined in Table 4-63: TCPCLv4 Message Type Byte Codes. For Transfer Refusal, this field is set to XFER_REFUSE.

Reason Code is an 8-bit unsigned integer value indicating the reason the transfer is being refused. Reason Code values are defined in Table 4-67: TCPCLv4 Transfer Refusal Reason Codes.

Transfer ID is a 64-bit unsigned integer value identifying the transfer being refused.

7.1.4.7 TCPCLv4 Message Rejection

The TCPCLv4 Message Rejection is composed of a header, reason code, and rejected message header. Table 7-34: TCPCLv4 Message Rejection defines the structure of TCPCLv4 Message Rejection.

Table 7-34: TCPCLv4 Message Rejection

Bit	0	1	2	3	4	5	6	7
Octet 0	message header							
Octet 1	reason code							
Octet 2	rejected message header							

Message Header is an 8-bit integer value indicating the type of message that follows. Message Header values are defined in Table 4-63: TCPCLv4 Message Type Byte Codes. For Message Rejection, this field is set to MSG_REJECT.

Reason Code is an 8-bit unsigned integer value indicating the reason the message is being rejected. Reason Code values are defined in Table 4-61: TCPCLv4 Message Rejection Reason Codes.

Rejected Message Header is an 8-bit integer value indicating the message being rejected. This value is a copy of the Message Header from the rejected message.

7.1.4.8 TCPCLv4 Session Upkeep

The TCPCLv4 Session Upkeep is composed of a header with no additional data. Table 7-35: TCPCLv4 Session Upkeep defines the structure of TCPCLv4 Session Upkeep.

Table 7-35: TCPCLv4 Session Upkeep

Bit	0	1	2	3	4	5	6	7
Octet 0	message header							

Message Header is an 8-bit integer value indicating the type of message that follows. Message Header values are defined in Table 4-63: TCPCLv4 Message Type Byte Codes. For Session Upkeep, this field is set to KEEPALIVE.

7.1.4.9 TCPCLv4 Session Termination Message

The TCPCLv4 Session Termination message is composed of a header with no additional data. Table 7-36: TCPCLv4 Session Upkeep defines the structure of TCPCLv4 Session Upkeep.

Table 7-36: TCPCLv4 Session Upkeep

Bit	0	1	2	3	4	5	6	7
Octet 0	message header							
Octet 1	message flags							
Octet 2	reason code							

Message Header is an 8-bit integer value indicating the type of message that follows. Message Header values are defined in Table 4-63: TCPCLv4 Message Type Byte Codes. For Session Termination, this field is set to SESS_TERM.

Message Flags is a bit-packed 8-bit structure. Table 7-37: TCPCLv4 Session Termination – Message Flags defines the structure of Message Flags.

Table 7-37: TCPCLv4 Session Termination – Message Flags

Mnemonic	**Bit Position**	**Description**
RESERVED	0 (MSB)	Reserved for future use
RESERVED	1	Reserved for future use
RESERVED	2	Reserved for future use
RESERVED	3	Reserved for future use
RESERVED	4	Reserved for future use
RESERVED	5	Reserved for future use
RESERVED	6	Reserved for future use
REPLY	7 (LSB)	This message is an acknowledgment of an earlier Session Termination message

Reason Code is an 8-bit unsigned integer value indicating the reason the session is being terminated. Reason Code values are defined in Table 4-65: TCPCLv4 Session Termination Reason Codes.

7.1.5 UDPCL

UDP Convergence Layer (UDPCL) bridges the gap between the DTN Bundle Protocol at the application layer and UDP at the transport layer.

UDPCL messages have the following formats:

- UDP Bundle Data

7.1.5.1 UDPCL Bundle Data

UDPCL Bundle Data is composed of bundle data. Table 7-38: UDPCL Bundle Data defines the structure of UDPCL Bundle Data.

Table 7-38: UDPCL Bundle Data

Bit	0	1	2	3	4	5	6	7
Octet 0	bundle data (byte 0)							
…	…							
Octet X	bundle data (byte X)							

Bundle Data is a payload of variable length.

7.2 Inter-CSC I/O

The section describes data produced by individual HDTN Computer Software Components (CSCs) for consumption by one or more other HDTN CSCs.

7.2.1 Egress Inputs

Table 7-39: Egress Inputs captures the data produced by other HDTN CSCs for consumption by the Egress CSC.

Table 7-39: Egress Inputs

Mnemonic	Description	Source	Type
Bundle	Bundle	Ingress CSC, Storage CSC	Buffer
ToEgressHdr	To egress header	Ingress CSC, Storage CSC	To Egress Header
IreleaseChangeHdr	Release change header	Router CSC	Release Change Header
RouteUpdateHdr	Route update header	Router CSC	Route Update Header

7.2.2 Egress Outputs

Table 7-40: Egress Outputs captures the data produced by the Egress CSC for consumption by other HDTN CSCs.

Table 7-40: Egress Outputs

Mnemonic	Description	Destination	Type
EgressAckHdr	Egress Acknowledge Header	Ingress CSC, Storage CSC	Egress Acknowledge Header
Opportunistic Bundle	Opportunistic Bundle	Ingress CSC	Buffer
AllOutductCapabilitiesTelemetry_t	All Outduct Capabilities Telemetry	Ingress CSC, Router CSC, Storage CSC	All Outduct Capabilities Telemetry
LinkStatusHdr	Link Status Header	Router CSC	Link Status Header

7.2.3 Ingress Inputs

Table 7-41: Ingress Inputs captures the data produced by other HDTN CSCs for consumption by the Ingress CSC.

Table 7-41: Ingress Inputs

Mnemonic	Description	Source	Type
AllOutductCapabilitiesTelemetry_t	All outduct capabilities telemetry	Egress CSC	All Outduct Capabilities Telemetry
EgressAckHdr	Egress acknowledge header	Egress CSC	Egress Acknowledge Header
Opportunistic Bundle	Opportunistic bundle	Egress CSC	Buffer
IreleaseChangeHdr	Release change header	Router CSC	Release Change Header
StorageAckHdr	Storage acknowledge header	Storage CSC	Storage Acknowledge Header

7.2.4 Ingress Outputs

Table 7-42: Ingress Outputs captures the data produced by the Ingress CSC for consumption by other HDTN CSCs.

Table 7-42: Ingress Outputs

Mnemonic	Description	Destination	Type
Bundle	Bundle	Egress CSC, Storage CSC	Buffer
ToEgressHdr	To egress header	Egress CSC	To Egress Header
ToStorageHdr	To storage header	Storage CSC	To Storage Header

7.2.5 Router Inputs

Table 7-43: Router Inputs captures the data produced by other HDTN CSCs for consumption by the Router CSC.

Table 7-43: Router Inputs

Mnemonic	Description	Source	Type
AllOutductCapabilitiesTelemetry_t	All outduct capabilities telemetry	Egress CSC	All Outduct Capabilities Telemetry
LinkStatusHdr	Link status header	Egress CSC	Link Status Header
DepletedStorageReportHdr	Depleted storage report header	Storage CSC	Depleted Storage Report Header

7.2.6 Router Outputs

Table 7-44: Router Outputs captures the data produced by the Router CSC for consumption by other HDTN CSCs.

Table 7-44: Router Outputs

Mnemonic	Description	Destination	Type
IreleaseChangeHdr	Release change header	Egress CSC, Ingress CSC, Storage CSC	Release Change Header
RouteUpdateHdr	Route update header	Egress CSC	Route Update Header

7.2.7 Storage Inputs

Table 7-45: Storage Inputs captures the data produced by other HDTN CSCs for consumption by the Storage CSC.

Table 7-45: Storage Inputs

Mnemonic	Description	Source	Type
AllOutductCapabilitiesTelemetry_t	All outduct capabilities telemetry	Egress CSC	All Outduct Capabilities Telemetry
Bundle	Bundle	Egress CSC, Ingress CSC	Buffer
EgressAckHdr	Egress acknowledge header	Egress CSC	Egress Acknowledge Header
ToStorageHdr	To storage header	Ingress CSC	To Storage Header
IreleaseChangeHdr	Release change header	Router CSC	Release Change Header

7.2.8 Storage Outputs

Table 7-46: Storage Outputs captures the data produced by the Storage CSC for consumption by other HDTN CSCs.

Table 7-46: Storage Outputs

Mnemonic	Description	Destination	Type
ToEgressHdr	To egress header	Egress CSC	To Egress Header
StorageAckHdr	Storage acknowledge header	Ingress CSC	Storage Acknowledge Header
DepletedStorageReportHdr	Depleted storage report header	Router CSC	Depleted Storage Report Header

8.0 Application Programming Interface Data

This section defines the purpose, function, format, parameters, and any restrictions for API calls processed by the software.

More information on the use of API calls can be found in HDTN-USER-035 High-Rate Delay Tolerant Networking (HDTN) User Guide.

In the API calls below, the following formatting is used to explain syntax:

- `plain_text` – This text needs to be entered as it appears when making the API call.
- *`italicized_text`* – This text indicates a variable. Substitute an appropriate value for this text when making the API call.

In the returns for each API call below, the following formatting is used to explain syntax:

- `plain_text` – This text is returned as it appears.
- *`italicized_text`* – This text indicates a variable. Appropriate values for this text are substituted in the returned text.

8.1 Get BPSec Config

The Get BPSec Config call is used to obtain HDTN BP Security configuration information.

```
"apiCall": "get_bpsec_config"
```

The Get BPSec Config call has no required or optional input parameters.

Table 8-1: Get BPSec Config Return Parameters defines the return parameters for the Get BPSec Config call.

Table 8-1: Get BPSec Config Return Parameters

Mnemonic	Description	Type	Min	Max	Units
N/A	Current BPSec configuration	JSON string	N/A	N/A	N/A

8.2 Get Expiring Storage

The Get Expiring Storage call is used to obtain HDTN expiring storage statistics.

```
"apiCall": "get_expiring_storage",
"priority": priority,
"thresholdSecondsFromNow": thresholdSecondsFromNow
```

Table 8-2: Get Expiring Storage Input Parameters defines the input parameters for the Get Expiring Storage call.

Table 8-2: Get Expiring Storage Input Parameters

Mnemonic	Description	Type	Min	Max	Units	Required?
priority	The DTN priority to use for filtering the bundles that are counted	uint64	0	2	N/A	yes
thresholdSecondFromNow	The number of seconds from now to look for expiring bundles	uint64	0	UINT64_MAX	sec	yes

Table 8-3: Get Expiring Storage Return Parameters defines the return parameters for the Get Expiring Storage call.

Table 8-3: Get Expiring Storage Return Parameters

Mnemonic	Description	Type	Min	Max	Units
N/A	HDTN expiring storage statistics	JSON string	N/A	N/A	N/A

Figure 8-1: HDTN Expiring Storage Statistics String provides an example of the returned JSON string.

```
{
  "priority": priority,
  "thresholdSecondsSinceStartOfYear2000":
thresholdSecondsSinceStartOfYear2000,
  "nodesExpiringBeforeThresholdCount": [
    {
      "nodeId": nodeId,
      "bundleCount": bundleCount,
      "totalBundleBytes": totalBundleBytes
    }
  ]
}
```

Figure 8-1: HDTN Expiring Storage Statistics String

8.3 Get HDTN Config

The Get HDTN Config call is used to obtain the currently active configuration information for HDTN.

```
"apiCall": "get_hdtn_config"
```

The Get HDTN Config call has no required or optional input parameters.

Table 8-4: Get HDTN Config Return Parameters defines the return parameters for the Get HDTN Config call.

Table 8-4: Get HDTN Config Return Parameters

Mnemonic	Description	Type	Min	Max	Units
N/A	Current HDTN configuration	JSON string	N/A	N/A	N/A

Figure 8-2: Current HDTN Configuration String provides an example of the returned JSON string.

```
{
  "hdtnConfigName": "hdtnConfigName",
  "userInterfaceOn": userInterfaceOn,
  "mySchemeName": "mySchemeName",
  "myNodeId": myNodeId,
  "myBpEchoServiceId": myBpEchoServiceId,
  "myCustodialSsp": "myCustodialSsp",
  "myCustodialServiceId": myCustodialServiceId,
  "myRouterServiceId": myRouterServiceId,
  "isAcsAware": isAcsAware,
  "acsMaxFillsPerAcsPacket": acsMaxFillsPerAcsPacket,
  "acsSendPeriodMilliseconds": acsSendPeriodMilliseconds,
  "retransmitBundleAfterNoCustodySignalMilliseconds":
retransmitBundleAfterNoCustodySignalMilliseconds,
  "maxBundleSizeBytes": maxBundleSizeBytes,
  "maxIngressBundleWaitOnEgressMilliseconds":
maxIngressBundleWaitOnEgressMilliseconds,
  "bufferRxToStorageOnLinkUpSaturation": bufferRxToStorageOnLinkUpSaturation,
  "maxLtpReceiveUdpPacketSizeBytes": maxLtpReceiveUdpPacketSizeBytes,
  "neighborDepletedStorageDelaySeconds": neighborDepletedStorageDelaySeconds,
  "fragmentBundlesLargerThanBytes": fragmentBundlesLargerThanBytes,
  "enforceBundlePriority": enforceBundlePriority,
  "zmqBoundRouterPubSubPortPath": zmqBoundRouterPubSubPortPath,
  "zmqBoundTelemApiPortPath": zmqBoundTelemApiPortPath,
  "inductsConfig": {
    "inductConfigName": "inductConfigName",
    "inductVector": [
      {
        "name": "name",
        "convergenceLayer": "convergenceLayer",
        "boundPort": boundPort,
        "numRxCircularBufferElements": numRxCircularBufferElements,
        "thisLtpEngineId": thisLtpEngineId,
        "remoteLtpEngineId": remoteLtpEngineId,
        "ltpReportSegmentMtu": ltpReportSegmentMtu,
        "oneWayLightTimeMs": oneWayLightTimeMs,
        "oneWayMarginTimeMs": oneWayMarginTimeMs,
        "clientServiceId": clientServiceId,
        "preallocatedRedDataBytes": preallocatedRedDataBytes,
        "ltpMaxRetriesPerSerialNumber": ltpMaxRetriesPerSerialNumber,
        "ltpRandomNumberSizeBits": ltpRandomNumberSizeBits,
        "ltpRemoteUdpHostname": "ltpRemoteUdpHostname",
        "ltpRemoteUdpPort": ltpRemoteUdpPort,
        "ltpRxDataSegmentSessionNumberRecreationPreventerHistorySize":
ltpRxDataSegmentSessionNumberRecreationPreventerHistorySize,
        "ltpMaxExpectedSimultaneousSessions": ltpMaxExpectedSimultaneousSessions,
        "ltpMaxUdpPacketsToSendPerSystemCall": ltpMaxUdpPacketsToSendPerSystemCall,
        "delaySendingOfReportSegmentsTimeMsOrZeroToDisable":
delaySendingOfReportSegmentsTimeMsOrZeroToDisable,
        "keepActiveSessionDataOnDisk": keepActiveSessionDataOnDisk,
        "activeSessionDataOnDiskNewFileDurationMs":
activeSessionDataOnDiskNewFileDurationMs,
        "activeSessionDataOnDiskDirectory": "activeSessionDataOnDiskDirectory"
      }
    ]
  },
  "outductsConfig": {
```

```
    "outductConfigName": "outductConfigName",
    "outductVector": [
      {
        "name": "name",
        "convergenceLayer": "convergenceLayer",
        "nextHopNodeId": nextHopNodeId,
        "remoteHostname": "remoteHostname",
        "remotePort": remotePort,
        "maxNumberOfBundlesInPipeline": maxNumberOfBundlesInPipeline,
        "maxSumOfBundleBytesInPipeline": maxSumOfBundleBytesInPipeline,
        "thisLtpEngineId": thisLtpEngineId,
        "remoteLtpEngineId": remoteLtpEngineId,
        "ltpDataSegmentMtu": ltpDataSegmentMtu,
        "oneWayLightTimeMs": oneWayLightTimeMs,
        "oneWayMarginTimeMs": oneWayMarginTimeMs,
        "clientServiceId": clientServiceId,
        "numRxCircularBufferElements": numRxCircularBufferElements,
        "ltpMaxRetriesPerSerialNumber": ltpMaxRetriesPerSerialNumber,
        "ltpCheckpointEveryNthDataSegment": ltpCheckpointEveryNthDataSegment,
        "ltpRandomNumberSizeBits": ltpRandomNumberSizeBits,
        "ltpSenderBoundPort": ltpSenderBoundPort,
        "ltpMaxUdpPacketsToSendPerSystemCall": ltpMaxUdpPacketsToSendPerSystemCall,
        "ltpSenderPingSecondsOrZeroToDisable": ltpSenderPingSecondsOrZeroToDisable,
        "delaySendingOfDataSegmentsTimeMsOrZeroToDisable":
delaySendingOfDataSegmentsTimeMsOrZeroToDisable,
        "keepActiveSessionDataOnDisk": keepActiveSessionDataOnDisk,
        "activeSessionDataOnDiskNewFileDurationMs":
activeSessionDataOnDiskNewFileDurationMs,
        "activeSessionDataOnDiskDirectory": "activeSessionDataOnDiskDirectory",
        "rateLimitPrecisionMicroSec": rateLimitPrecisionMicroSec
      }
    ]
  },
  "storageConfig": {
    "storageImplementation": "storageImplementation",
    "tryToRestoreFromDisk": tryToRestoreFromDisk,
    "autoDeleteFilesOnExit": autoDeleteFilesOnExit,
    "totalStorageCapacityBytes": totalStorageCapacityBytes,
    "storageDeletionPolicy": "storageDeletionPolicy",
    "storageDiskConfigVector": [
      {
        "name": "name",
        "storeFilePath": "storeFilePath"
      },
      {
        "name": "name",
        "storeFilePath": "storeFilePath"
      }
    ]
  }
}
```

Figure 8-2: Current HDTN Configuration String

8.4 Get HDTN Version

The Get HDTN Version call is used to obtain the current HDTN version number.

```
"apiCall": "get_hdtn_version"
```

The Get HDTN Version call has no required or optional input parameters.

Table 8-5: Get HDTN Version Return Parameters defines the return parameters for the Get HDTN Version call.

Table 8-5: Get HDTN Version Return Parameters

Mnemonic	Description	Type	Min	Max	Units
N/A	Current HDTN version	JSON string	N/A	N/A	N/A

Figure 8-3: Current HDTN Version String provides an example of the returned JSON string.

```
{
  "version": "version"
}
```

Figure 8-3: Current HDTN Version String

8.5 Get Inducts

The Get Inducts call is used to obtain HDTN inducts statistics.

```
"apiCall": "get_inducts"
```

The Get Inducts call has no required or optional input parameters.

Table 8-6: Get HDTN Configuration Return Parameters defines the return parameters for the Get Inducts call.

Table 8-6: Get HDTN Configuration Return Parameters

Mnemonic	Description	Type	Min	Max	Units
N/A	All induct telemetry	JSON string	N/A	N/A	N/A

Figure 8-4: All Induct Telemetry String provides an example of the returned JSON string.

```
{
  "timestampMilliseconds": timestampMilliseconds,
  "bundleCountEgress": bundleCountEgress,
  "bundleCountStorage": bundleCountStorage,
  "bundleByteCountEgress": bundleByteCountEgress,
  "bundleByteCountStorage": bundleByteCountStorage,
  "allInducts": [
    {
      "convergenceLayer": "convergenceLayer",
```

```
      "inductConnections": [
        {
          "connectionName": "connectionName",
          "inputName": "inputName",
          "totalBundlesReceived": totalBundlesReceived,
          "totalBundleBytesReceived": totalBundleBytesReceived,
          "numReportSegmentTimerExpiredCallbacks":
numReportSegmentTimerExpiredCallbacks,
          "numReportSegmentsUnableToBeIssued": numReportSegmentsUnableToBeIssued,
          "numReportSegmentsTooLargeAndNeedingSplit":
numReportSegmentsTooLargeAndNeedingSplit,
          "numReportSegmentsCreatedViaSplit": numReportSegmentsCreatedViaSplit,
          "numGapsFilledByOutOfOrderDataSegments":
numGapsFilledByOutOfOrderDataSegments,
          "numDelayedFullyClaimedPrimaryReportSegmentsSent":
numDelayedFullyClaimedPrimaryReportSegmentsSent,
          "numDelayedFullyClaimedSecondaryReportSegmentsSent":
numDelayedFullyClaimedSecondaryReportSegmentsSent,
          "numDelayedPartiallyClaimedPrimaryReportSegmentsSent":
numDelayedPartiallyClaimedPrimaryReportSegmentsSent,
          "numDelayedPartiallyClaimedSecondaryReportSegmentsSent":
numDelayedPartiallyClaimedSecondaryReportSegmentsSent,
          "totalCancelSegmentsStarted": totalCancelSegmentsStarted,
          "totalCancelSegmentSendRetries": totalCancelSegmentSendRetries,
          "totalCancelSegmentsFailedToSend": totalCancelSegmentsFailedToSend,
          "totalCancelSegmentsAcknowledged": totalCancelSegmentsAcknowledged,
          "numRxSessionsCancelledBySender": numRxSessionsCancelledBySender,
          "numStagnantRxSessionsDeleted": numStagnantRxSessionsDeleted,
          "countUdpPacketsSent": countUdpPacketsSent,
          "countRxUdpCircularBufferOverruns": countRxUdpCircularBufferOverruns,
          "countTxUdpPacketsLimitedByRate": countTxUdpPacketsLimitedByRate
        }
      ]
    }
  ]
}
```

Figure 8-4: All Induct Telemetry String

8.6 Get Outducts

The Get Outducts call is used to obtain HDTN outducts statistics.

```
"apiCall": "get_outducts"
```

The Get Outducts call has no required or optional input parameters.

Table 8-7: Get Outducts Return Parameters defines the return parameters for the Get Outducts call.

Table 8-7: Get Outducts Return Parameters

Mnemonic	Description	Type	Min	Max	Units
N/A	All outduct telemetry	JSON string	N/A	N/A	N/A

Figure 8-5: All Outduct Telemetry String provides an example of the returned JSON string.

```
{
  "timestampMilliseconds": timestampMilliseconds,
  "totalBundlesGivenToOutducts": totalBundlesGivenToOutducts,
  "totalBundleBytesGivenToOutducts": totalBundleBytesGivenToOutducts,
  "totalTcpclBundlesReceived": totalTcpclBundlesReceived,
  "totalTcpclBundleBytesReceived": totalTcpclBundleBytesReceived,
  "totalStorageToIngressOpportunisticBundles":
totalStorageToIngressOpportunisticBundles,
  "totalStorageToIngressOpportunisticBundleBytes":
totalStorageToIngressOpportunisticBundleBytes,
  "totalBundlesSuccessfullySent": totalBundlesSuccessfullySent,
  "totalBundleBytesSuccessfullySent": totalBundleBytesSuccessfullySent,
  "allOutducts": [
    {
      "convergenceLayer": "convergenceLayer",
      "totalBundlesAcked": totalBundlesAcked,
      "totalBundleBytesAcked": totalBundleBytesAcked,
      "totalBundlesSent": totalBundlesSent,
      "totalBundleBytesSent": totalBundleBytesSent,
      "totalBundlesFailedToSend": totalBundlesFailedToSend,
      "linkIsUpPhysically": linkIsUpPhysically,
      "linkIsUpPerTimeSchedule": linkIsUpPerTimeSchedule,
      "numCheckpointsExpired": numCheckpointsExpired,
      "numDiscretionaryCheckpointsNotResent": numDiscretionaryCheckpointsNotResent,
      "numDeletedFullyClaimedPendingReports": numDeletedFullyClaimedPendingReports,
      "totalCancelSegmentsStarted": totalCancelSegmentsStarted,
      "totalCancelSegmentSendRetries": totalCancelSegmentSendRetries,
      "totalCancelSegmentsFailedToSend": totalCancelSegmentsFailedToSend,
      "totalCancelSegmentsAcknowledged": totalCancelSegmentsAcknowledged,
      "totalPingsStarted": totalPingsStarted,
      "totalPingRetries": totalPingRetries,
      "totalPingsFailedToSend": totalPingsFailedToSend,
      "totalPingsAcknowledged": totalPingsAcknowledged,
      "numTxSessionsReturnedToStorage": numTxSessionsReturnedToStorage,
      "numTxSessionsCancelledByReceiver": numTxSessionsCancelledByReceiver,
      "countUdpPacketsSent": countUdpPacketsSent,
      "countRxUdpCircularBufferOverruns": countRxUdpCircularBufferOverruns,
      "countTxUdpPacketsLimitedByRate": countTxUdpPacketsLimitedByRate
    }
  ]
}
```

Figure 8-5: All Outduct Telemetry String

8.7 Get Storage

The Get Storage call is used to obtain HDTN storage statistics.

```
"apiCall": "get_storage"
```

The Get Storage call has no required or optional input parameters.

Table 8-8: Get Storage Return Parameters defines the return parameters for the Get Storage call.

Table 8-8: Get Storage Return Parameters

Mnemonic	Description	Type	Min	Max	Units
N/A	Storage telemetry	JSON string	N/A	N/A	N/A

Figure 8-6: Storage Telemetry String provides an example of the returned JSON string.

```
{
  "timestampMilliseconds": timestampMilliseconds,
  "totalBundlesErasedFromStorageNoCustodyTransfer":
totalBundlesErasedFromStorageNoCustodyTransfer,
  "totalBundlesErasedFromStorageWithCustodyTransfer":
totalBundlesErasedFromStorageWithCustodyTransfer,
  "totalBundlesErasedFromStorageBecauseExpired":
totalBundlesErasedFromStorageBecauseExpired,
  "totalBundlesRewrittenToStorageFromFailedEgressSend":
totalBundlesRewrittenToStorageFromFailedEgressSend,
  "totalBundlesSentToEgressFromStorageReadFromDisk":
totalBundlesSentToEgressFromStorageReadFromDisk,
  "totalBundleBytesSentToEgressFromStorageReadFromDisk":
totalBundleBytesSentToEgressFromStorageReadFromDisk,
  "totalBundlesSentToEgressFromStorageForwardCutThrough":
totalBundlesSentToEgressFromStorageForwardCutThrough,
  "totalBundleBytesSentToEgressFromStorageForwardCutThrough":
totalBundleBytesSentToEgressFromStorageForwardCutThrough,
  "numRfc5050CustodyTransfers": numRfc5050CustodyTransfers,
  "numAcsCustodyTransfers": numAcsCustodyTransfers,
  "numAcsPacketsReceived": numAcsPacketsReceived,
  "numBundlesOnDisk": numBundlesOnDisk,
  "numBundleBytesOnDisk": numBundleBytesOnDisk,
  "totalBundleWriteOperationsToDisk": totalBundleWriteOperationsToDisk,
  "totalBundleByteWriteOperationsToDisk": totalBundleByteWriteOperationsToDisk,
  "totalBundleEraseOperationsFromDisk": totalBundleEraseOperationsFromDisk,
  "totalBundleByteEraseOperationsFromDisk": totalBundleByteEraseOperationsFromDisk,
  "usedSpaceBytes": usedSpaceBytes,
  "freeSpaceBytes": freeSpaceBytes
}
```

Figure 8-6: Storage Telemetry String

8.8 Ping

The Ping call is used to ping a specific HDTN node service.

```
"apiCall": "ping",
"bpVersion": bpVersion,
"nodeId": nodeId,
"serviceId": serviceId
```

Table 8-9: Ping Inputs defines the input parameters for the Ping call.

Table 8-9: Ping Inputs

Mnemonic	Description	Type	Min	Max	Units	Required?
bpVersion	The bundle protocol version to use for generating the ping message	uint64	6	7	N/A	yes
nodeId	The node ID of the destination for the ping message	uint64	0	UINT64_MAX	N/A	yes
serviceId	The service number of the destination application for the ping message	uint64	0	UINT64_MAX	N/A	yes

Table 8-10: Ping Return Parameters defines the return parameters for the Ping call.

Table 8-10: Ping Return Parameters

Mnemonic	Description	Type	Min	Max	Units
N/A	Ping reply	JSON string	N/A	N/A	N/A

Figure 8-7: Ping Reply String provides an example of the returned JSON string.

```
{
  "success": success,
  "message": "message"
}
```

Figure 8-7: Ping Reply String

8.9 Set Link Down

The Set Link Down call is used to take a link down from the outductVector.

```
"apiCall": "set_link_down",
"outductIndex": outductIndex
```

Table 8-11: Set Link Down Inputs defines the input parameters for the Set Link Down call.

Table 8-11: Set Link Down Inputs

Mnemonic	Description	Type	Min	Max	Units	Required?
outductIndex	The index of the link to take down in the HDTN outduct vector	uint64	0	lesser of: (number of outducts – 1) OR UINT64_MAX	N/A	yes

Table 8-12: Set Link Down Return Parameters defines the return parameters for the Set Link Down call.

Table 8-12: Set Link Down Return Parameters

Mnemonic	Description	Type	Min	Max	Units
N/A	Set link down status	JSON string	N/A	N/A	N/A

Figure 8-8: Set Link Down Status String provides an example of the returned JSON string.

```
{
  "success": success,
  "message": "message"
}
```

Figure 8-8: Set Link Down Status String

8.10 Set Link Up

The Set Link Up call is used to take a link down from the outductVector.

```
"apiCall": "set_link_up",
"outductIndex": outductIndex
```

Table 8-13: Set Link Up Inputs defines the input parameters for the Set Link Up call.

Table 8-13: Set Link Up Inputs

Mnemonic	Description	Type	Min	Max	Units	Required?
outductIndex	The index of the link to bring up in the HDTN outduct vector	uint64	0	lesser of: (number of outducts – 1) OR UINT64_MAX	N/A	yes

Table 8-14: Set Link Up Return Parameters defines the return parameters for the Set Link Up call.

Table 8-14: Set Link Up Return Parameters

Mnemonic	Description	Type	Min	Max	Units
N/A	Set link up status	JSON string	N/A	N/A	N/A

Figure 8-9: Set Link Up Status String provides an example of the returned JSON string.

```
{
  "success": success,
  "message": "message"
}
```

Figure 8-9: Set Link Up Status String

8.11 Set Max Send Rate

The Set Max Send Rate call is used to set the maximum send rate (in bits per second) for a specific HDTN outduct.

```
"apiCall": "set_max_send_rate",
"rateBitsPerSec": rateBitsPerSec,
"outduct": outduct
```

Table 8-15: Set Max Send Rate Input Parameters defines the input parameters for the Set Max Send Rate call.

Table 8-15: Set Max Send Rate Input Parameters

Mnemonic	Description	Type	Min	Max	Units	Required?
rateBitsPerSec	The maximum rate in bits per second; when the input is 0, there is no upper limit imposed on the maximum rate.	uint64	0	UINT64_MAX	bits/sec	yes
outduct	Index number of the HDTN outduct on which to set the maximum send rate	uint64	0	lesser of: (number of outducts – 1) OR UINT64_MAX	index	yes

Table 8-16: Set Max Send Rate Return Parameters defines the return parameters for the Set Max Send Rate call.

Table 8-16: Set Max Send Rate Return Parameters

Mnemonic	Description	Type	Min	Max	Units
N/A	Set max send rate status	JSON string	N/A	N/A	N/A

Figure 8-10: Set Max Send Rate Status String provides an example of the returned JSON string.

```
{
  "success": success,
  "message": "message"
}
```

Figure 8-10: Set Max Send Rate Status String

8.12 Upload Contact Plan

The Upload Contact Plan call is used to upload a HDTN contact plan.

```
"apiCall": "upload_contact_plan",
"contactPlanJson": contactPlanJson
```

Table 8-17: Upload Contact Plan Input Parameters defines the input parameters for the Upload Contact Plan call.

Table 8-17: Upload Contact Plan Input Parameters

Mnemonic	Description	Type	Min	Max	Units	Required?
contactPlanJson	JSON-encoded string specifying the new contact plan (See Appendix E Example: Contact Plan Configuration File)	JSON string	N/A	N/A	N/A	yes

Table 8-18: Upload Contact Plan Return Parameters defines the return parameters for the Upload Contact Plan call.

Table 8-18: Upload Contact Plan Return Parameters

Mnemonic	Description	Type	Min	Max	Units
N/A	Upload contact plan status	JSON string	N/A	N/A	N/A

Figure 8-11: Upload Contact Plan Status String provides an example of the returned JSON string.

```
{
  "success": success,
  "message": "message"
}
```

Figure 8-11: Upload Contact Plan Status String

9.0 File System Structure

HDTN software relies on path variables defined in the run script to locate files in the local file system.

An example of a HDTN run script is included in Appendix C Example: HDTN Run Script.

9.1 Supported Settings

The following file system paths are configurable:

- Configuration files location (config_files)
- HDTN configuration location (hdtn_config)
- BPSec configuration location (bpsec_config)
- BPSink configuration location (sink_config)
- BPGen configuration location (gen_config)

10.0 Telemetry Data

This section defines formats and units of measure of data collected from remote sources for transmission to a central location.

Table 10-1: Available Telemetry Data defines the telemetry data available to be recorded in telemetry data log files:

Table 10-1: Available Telemetry Data

Mnemonic	Description	Source	Type	Min	Max	Units
ingress_data_rate_mbps	Ingress data receive rate	telemetry	double	0	DBL_MAX	Mbits/sec
ingress_total_bytes_sent	Total size of data sent by ingress to egress and storage	telemetry	uint64	0	UINT64_MAX	bytes
ingress_bytes_sent_egress	Total size of data sent by ingress to egress	telemetry	uint64	0	UINT64_MAX	bytes
ingress_bytes_sent_storage	Total size of data sent by ingress to storage	telemetry	uint64	0	UINT64_MAX	bytes
storage_used_space_bytes	Total storage space used	telemetry	uint64	0	UINT64_MAX	bytes
storage_free_space_bytes	Total storage space free	telemetry	uint64	0	UINT64_MAX	bytes
storage_bundle_bytes_on_disk	Total size of bundles stored on disk	telemetry	uint64	0	UINT64_MAX	bytes
storage_bundles_erased	Total number of bundles erased (with and without custody transfer)	telemetry	uint64	0	UINT64_MAX	N/A
storage_bundles_rewritten_from_failed_egress_send	Total number of bundles erased from failed egress send	telemetry	uint64	0	UINT64_MAX	N/A
storage_bytes_sent_to_egress_cutthrough	Total size of data sent to egress cut through	telemetry	uint64	0	UINT64_MAX	bytes
storage_bytes_sent_to_egress_from_disk	Total size of data sent to egress from disk	telemetry	uint64	0	UINT64_MAX	bytes
egress_data_rate_mbps	Egress data transmit rate	telemetry	double	0	DBL_MAX	Mbits/sec
egress_total_bytes_sent_success	Total size of bundles successfully transmitted	telemetry	uint64	0	UINT64_MAX	bytes
egress_total_bytes_attempted	Total size of bundle transmissions attempted	telemetry	uint64	0	UINT64_MAX	bytes

11.0 Recorded Data

This section defines data collected and recorded to a storage system.

11.1 Bundle Segment Storage

Bundle Segment Storage data is composed of bundle size, next segment ID, assigned custody ID, and bundle data. Table 11-1: Stored Bundle Segment Structure defines the structure of stored bundle segments.

Table 11-1: Stored Bundle Segment Structure

Bit	0	1	2	3	4	5	6	7
Octet 0	bundle size (byte 0)							
…	…							
Octet 7	bundle size (byte 7)							
Octet 8	next segment ID (byte 0)							
…	…							
Octet 11	next segment ID (byte 3)							
Octet 12	assigned custody ID (byte 0)							
…	…							
Octet 19	assigned custody ID (byte 7)							
Octet 20	bundle data (byte 0)							
…	…							
Octet 20+X	bundle data (byte X)							

Bundle Size is a 64-bit integer value indicating the total size in bytes of the bundle before storage. Only the first logical segment in a bundle indicates the size of the bundle. All other logical segments in the bundle indicate a Bundle Size of UINT64_MAX. Bundles are “deleted” from storage by writing a value of UINT64_MAX to Bundle Size of the first logical segment.

Next Segment ID is a 32-bit integer value indicating the next logical segment in the bundle. A value of UINT32_MAX indicates that the logical segment is the last segment of a bundle.

Assigned Custody ID is a 64-bit integer value indicating the Custody ID originally assigned to this bundle.

Bundle Data is a variable length field containing the bundle fragment data. The length of this field for all but the last segment of the stored bundle is 4076 bytes. The length of this field for the last segment of a stored bundle can be determined using the following formula:

$$last\ segment\ length = (Bundle\ Size)\ mod\ (4076)$$

12.0 Configuration Data

This section defines the format and parameters of data necessary to configure the software.

12.1 Bundle Protocol Security Configuration

This section defines the format and parameters of data necessary to configure BPSec.

An example of a BPSec configuration file is included in Appendix D Example: BPSec Configuration File.

12.1.1 Supported Settings

Table 12-1: Security Context Parameter Configuration Fields defines available BPSec parameters and their valid values.

Table 12-1: Security Context Parameter Configuration Fields

Name	Required	Data Type	Input Type
"paramName"	TRUE	String	Select **Option Label** / **Option Value** "AES Variant" / "aesVariant" "IV Size Bytes" / "ivSizeBytes" "Key File" / "keyFile" "Security Block CRC" / "securityBlockCrc" "Scope Flags" / "scopeFlags" "Wrapped Key" / "wrappedKey"
"value"	TRUE	String	Text Field

Table 12-2: Policy Rules Configuration Fields defines available Policy Rules parameters and their valid values.

Table 12-2: Policy Rules Configuration Fields

Name	Required	Data Type	Input Type	Min	Max
"description"	TRUE	string	Text Field	N/A	N/A
"securityPolicyRuleId"	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
"securityRole"	TRUE	string	Select **Option Label** / **Option Value** "Acceptor" / "acceptor" "Source" / "source" "Verifier" / "verifier"	N/A	N/A
"securitySource"	TRUE	string	Text Field	N/A	N/A
"bundleSource"	FALSE	string array	Text Field	N/A	N/A
"bundleFinalDestination"	FALSE	string array	Text Field	N/A	N/A
"securityTargetBlockTypes"	FALSE	number array	Select **Option Label** / **Option Value** "Primary Implicit Zero" / 0 "Payload" / 1 "Previous Node" / 6 "Bundle Age" / 7 "Hop Count" / 10 "Integrity" / 11 "Confidentiality" / 12	0	12

Name	Required	Data Type	Input Type	Min	Max
“securityService”	TRUE	string	Select **Option Label** / **Option Value** “Confidentiality” / “confidentiality” “Integrity” / “integrity”	N/A	N/A
“securityContext”	TRUE	string	Select **Option Label** / **Option Value** “AES GCM” / “aesGcm” “HMAC SHA” / “hmacSha”	N/A	N/A
“securityContextParams”	FALSE	Security Context Parameter Configuration Fields object array	vector	N/A	N/A

Table 12-3: Security Operations Events Configuration Fields defines available Security Operations Events parameters and their valid values.

Table 12-3: Security Operations Events Configuration Fields

Name	Required	Data Type	Input Type	Min	Max
“eventId”	TRUE	string	Select **Option Label** / **Option Value** SOP Misconfigured At Verifier / “sopMisconfiguredAtVerifier” “SOP Missing At Verifier” / “sopMissingAtVerifier” “SOP Corrupted At Verifier” / “sopCorruptedAtVerifier” “SOP Misconfigured At Acceptor” / “sopMisconfiguredAtAcceptor” “SOP Missing At Acceptor” / “sopMissingAtAcceptor” “SOP Corrupted At Acceptor” / “sopCorruptedAtAcceptor”	N/A	N/A
“actions”	FALSE	string array	Select **Option Label** / **Option Value** “Remove Security Operation” / “removeSecurityOperation” “Remove Security Operation Target Block” / “removeSecurityOperationTargetBlock” “Remove All Security Target Operations” / “removeAllSecurityTargetOperations” “Fail Bundle Forwarding” / “failBundleForwarding” “Request Bundle Storage” / “requestBundleStorage” “Report Reason Code” / “reportReasonCode” “Override Security Target Block BPCF” / “overrideSecurityTargetBlockBpcf” “Override Security Block BPCF” / “overrideSecurityBlockBpcf”	N/A	N/A

Table 12-4: Security Failure Event Sets Configuration Fields defines available Security Failure Event Sets parameters and their valid values.

Table 12-4: Security Failure Event Sets Configuration Fields

Name	Required	Data Type	Input Type	Min	Max
“name”	TRUE	string	Text Field	N/A	N/A
“description”	TRUE	string	Text Field	N/A	N/A
“securityOperationEvents”	FALSE	Security Operations Events Configuration Fields object array	vector	N/A	N/A

Table 12-5: BPSec Configuration Fields defines available BPSec parameters and their valid values.

Table 12-5: BPSec Configuration Fields

Name	Required	Data Type	Input Type	Min	Max
“bpsecConfigName”	TRUE	string	Text Field	N/A	N/A
“policyRules”	FALSE	Policy Rules Configuration Fields “description” object array	vector	N/A	N/A
“securityFailureEventSets”	FALSE	Security Failure Event Sets Configuration Fields “name” object array	vector	N/A	N/A

12.2 Contact Plan Configuration

This section defines the format and parameters of data necessary to configure a contact plan.

An example of a contact plan configuration file is included in Appendix E Example: Contact Plan Configuration File

12.2.1 Supported Settings

Table 12-6: Contact Fields defines the structure of Contact Fields.

Table 12-6: Contact Fields

Name	Required	Data Type	Input Type	Min	Max
“contact”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“source”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“dest”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“startTime”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“endTime”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“rateBitsPerSec”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX

Table 12-7: Contact Plan Configuration Fields defines available Contact Plan parameters.

Table 12-7: Contact Plan Configuration Fields

Name	Required	Data Type	Input Type	Min	Max
“contacts”	FALSE	Contact Fields array	vector	N/A	N/A

12.3 Distributed Configuration

This section defines the format and parameters of data necessary to configure the HDTN modules for distributed operation.

An example of a distributed configuration file is included in Appendix F Example: Distributed Configuration File.

12.3.1 Supported Settings

Table 12-8: Distributed Configuration Fields defines available Distributed parameters and their valid values.

Table 12-8: Distributed Configuration Fields

Name	Required	Data Type	Input Type	Min	Max
"zmqIngressAddress"	TRUE	string	Text Field	N/A	N/A
"zmqEgressAddress"	TRUE	string	Text Field	N/A	N/A
"zmqStorageAddress"	TRUE	string	Text Field	N/A	N/A
"zmqRouterAddress"	TRUE	string	Text Field	N/A	N/A
"zmqBoundIngressToConnectingEgressPortPath"	TRUE	number	Text Field	0	65535
"zmqConnectingEgressToBoundIngressPortPath"	TRUE	number	Text Field	0	65535
"zmqBoundEgressToConnectingRouterPortPath"	TRUE	number	Text Field	0	65535
"zmqConnectingEgressBundlesOnlyToBoundIngressPortPath"	TRUE	number	Text Field	0	65535
"zmqBoundIngressToConnectingStoragePortPath"	TRUE	number	Text Field	0	65535
"zmqConnectingStorageToBoundIngressPortPath"	TRUE	number	Text Field	0	65535
"zmqConnectingStorageToBoundEgressPortPath"	TRUE	number	Text Field	0	65535
"zmqConnectingStorageToBoundRouterPortPath"	TRUE	number	Text Field	0	65535
"zmqBoundEgressToConnectingStoragePortPath"	TRUE	number	Text Field	0	65535
"zmqConnectingRouterToBoundEgressPortPath"	TRUE	number	Text Field	0	65535
"zmqConnectingTelemToFromBoundIngressPortPath"	TRUE	number	Text Field	0	65535
"zmqConnectingTelemToFromBoundEgressPortPath"	TRUE	number	Text Field	0	65535
"zmqConnectingTelemToFromBoundStoragePortPath"	TRUE	number	Text Field	0	65535
"zmqConnectingTelemToFromBoundRouterPortPath"	TRUE	number	Text Field	0	65535

12.4 High-Rate Delay Tolerant Networking Configuration

This section defines the format and parameters of data necessary to configure HDTN.

An example of a distributed configuration file is included in Appendix G Example: HDTN Configuration File.

12.4.1 Supported Settings

Table 12-9: Convergence Layer Options defines valid Convergence Layer options values.

Table 12-9: Convergence Layer Options

Option Label	Option Value
“STCP”	“stcp”
“UDP”	“udp”
“TCPLv3”	“tcpl_v3”
“TCPLv4”	“tcpl_v4”
“SLIP over UART”	“slip_over_uart”
“LTP over UDP”	“ltp_over_udp”
“LTP over Encap Local Stream”	“ltp_over_encap_local_stream”
“LTP over IPC”	“ltp_over_ipc”

Table 12-10: Induct Vector Fields defines available Induct Vector parameters and their valid values.

Table 12-10: Induct Vector Fields

Name	Required	Data Type	Input Type	Min	Max
“name”	TRUE	string	Text Field	N/A	N/A
“convergenceLayer”	TRUE	string	Select Convergence Layer Options	N/A	N/A

Table 12-11: Inducts Configuration Fields defines available Induct parameters and their valid values.

Table 12-11: Inducts Configuration Fields

Name	Required	Data Type	Input Type	Min	Max
“inductConfigName”	TRUE	string	Text Field	N/A	N/A
“inductVector”	FALSE	Induct Vector Fields “name” object array	vector	N/A	N/A

Table 12-12: Outduct Vector Fields defines available Outduct Vector parameters and their valid values.

Table 12-12: Outduct Vector Fields

Name	Required	Data Type	Input Type	Min	Max
“name”	TRUE	string	Text Field	N/A	N/A
“convergenceLayer”	TRUE	string	Select Convergence Layer Options	N/A	N/A
“nextHopNodeId”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“maxNumberOfBundlesInPipeline”	TRUE	number	Text Field	UINT32_MIN	UINT32_MAX
“maxSumOfBundleBytesInPipeline”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“bpEncapLocalSocketOrPipePath”	TRUE	string	Text Field	UINT64_MIN	UINT64_MAX

Table 12-13: Outducts Configuration Fields defines available Outducts parameters and their valid values.

Table 12-13: Outducts Configuration Fields

Name	Required	Data Type	Input Type	Min	Max
“outductConfigName”	TRUE	string	Text Field	N/A	N/A
“outductVector”	FALSE	Outduct Vector Fields “name” object array	vector	N/A	N/A

Table 12-14: Storage Disk Configuration Vector Fields defines available Storage Disk Configuration Vector parameters and their valid values.

Table 12-14: Storage Disk Configuration Vector Fields

Name	Required	Data Type	Input Type	Min	Max
“name”	TRUE	string	Text Field	N/A	N/A
“storageFilePath”	TRUE	string	Text Field	N/A	N/A

Table 12-15: Storage Configuration Fields defines available Storage parameters and their valid values.

Table 12-15: Storage Configuration Fields

Name	Required	Data Type	Input Type	Min	Max
“storageImplementation”	TRUE	string	Select **Option Label** / **Option Value** “ASIO Single Threaded” / “asio_single_threaded” “STDIO Multithreaded” / “stdio_multi_threaded”	N/A	N/A
“tryToRestoreFromDisk”	TRUE	Boolean	Switch	N/A	N/A
“autoDeleteFilesOnExit”	TRUE	Boolean	Switch	N/A	N/A
“totalStorageCapacityBytes”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“storageDeletionPolicy”	TRUE	string	Select **Option Label** / **Option Value** “Never” / “never” “On Expiration” / “on_expiration” “On Storage Full” / “on_storage_full”	N/A	N/A
“storageDiskConfigVector”	FALSE	Storage Disk Configuration Vector “name” object array	vector	N/A	N/A

Table 12-16: HDTN Configuration Fields defines available HDTN parameters and their valid values.

Table 12-16: HDTN Configuration Fields

Name	Required	Data Type	Input Type	Min	Max
“hdtnConfigName”	TRUE	string	Text Field	N/A	N/A
“userInterfaceOn”	TRUE	Boolean	Switch	N/A	N/A
“mySchemeName”	TRUE	string	Text Field	N/A	N/A
“myNodeId”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“myBpEchoServiceId”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“myCustodialSsp”	TRUE	string	Text Field	N/A	N/A
“myCustodialServiceId”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“myRouterServiceId”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“isAcsAware”	TRUE	Boolean	Switch	N/A	N/A
“acsMaxFillsPerAcsPacket”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“acsSendPeriodMilliseconds”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“retransmitBundleAfterNoCustodySignalMilliseconds”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“maxBundleSizeBytes”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“maxIngressBundleWaitOnEgressMilliseconds”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“bufferRxToStorageOnLinkUpSaturation”	TRUE	Boolean	Switch	N/A	N/A
“maxLtpReceiveUdpPacketSizeBytes”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“neighborDepletedStorageDelaySeconds”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“fragmentBundlesLargerThanBytes”	TRUE	number	Text Field	UINT64_MIN	UINT64_MAX
“enforceBundlePriority”	TRUE	Boolean	Switch	N/A	N/A
“zmqBoundRouterPubSubPortPath”	TRUE	number	Text Field	UINT16_MIN	UINT16_MAX
“zmqBoundTelemApiPortPath”	TRUE	number	Text Field	UINT16_MIN	UINT16_MAX
“inductsConfig”	TRUE	Inducts Configuration Fields object	vector	N/A	N/A
“outductsConfig”	TRUE	Outducts Configuration Fields object	vector	N/A	N/A
“storageConfig”	TRUE	Storage Configuration Fields object	vector	N/A	N/A

Appendix A Glossary and Acronyms

This section includes an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of terms and definitions needed to understand this document.

A.1 GLOSSARY

Table A-1: Definitions contains an alphabetized list of definitions for special terms used in the document; that is, the terms are used in a sense that differs from or is more specific than the common usage for such terms.

Table A-1: Definitions

Term	Definition
Activity	(1) any of the project components or research functions that are executed to deliver a product or service or provide support or insight to mature technologies (2) a set of tasks that describe the technical effort to accomplish a process and help generate expected outcomes
Agency	NASA
Architecture and Design	A description of the mission elements, their interfaces, their logical and physical layout, and the analysis of the design to determine expected performance and margins; includes System Design Synthesis, System Design Analysis, and System Design Validation products
Baseline	An agreed-to set of requirements, designs, or documents that will have changes controlled through a formal approval and monitoring process
Configuration Management	(1) a systematic process for establishing and maintaining control and evaluation of all changes to baseline documentation, products (Configuration Items), and subsequent changes to that documentation which defines the original scope of effort (2) the systematic control, identification, status accounting, and verification of all Configuration Items throughout their life cycle
Contractor	(per NPR 7123.1) an individual, partnership, company, corporation, association, or other service having a contract with the Agency for the design, development, manufacture, maintenance, modification, operation, or supply of items or services under the terms of a contract to a program or project within the scope of this NPR; research grantees, research contractors, and research subcontractors are excluded from this definition
Customer	an organization or individual that has requested a product and will receive the product to be delivered; may be an end user of the product, acquiring agent for the end user, or requestor of the work products from a technical effort

Term	Definition
Implementation Phase	part of the NASA management life cycle (defined in NPR 7120.5) where detailed design of system products is completed and products to be deployed are fabricated, assembled, integrated, and tested
LSS	GRC Flight Software Branch organizational code
Product	part of a system consisting of end products that perform operational functions and enabling products that perform life-cycle services related to the end product or a result of the technical efforts in the form of a work product
Program	strategic investment by a mission directorate that has defined goals, objectives, architecture, funding level, and management structure supporting one or more projects
Project	(1) specific investment having defined goals, objectives, requirements, life-cycle cost, a beginning, and an end; yields new or revised products or services that directly address NASA's strategic needs (2) unit of work performed in programs, projects, and activities
Shall	used to indicate a planned process that must be implemented; implementation must be verified
Should	used to indicate a goal that must be addressed by the plan; not formally verified
Specification	document that prescribes – in a complete, precise, verifiable manner – the requirements, design, behavior, or characteristics of a system or system component
Stakeholder	group or individual affected by or accountable for the outcome of an undertaking
SWE	generally refers to a specific NPR 7150.2 software engineering requirement
System	The combination of elements (hardware, software, equipment, facilities, personnel, processes, and procedures) that function together to produce the capability to meet a need
Validation	proof that the product accomplishes its intended purpose; may be determined by a combination of test, analysis, and demonstration
Verification	proof of compliance with specifications; may be determined by test, analysis, demonstration, and inspection
Waiver	documented agreement intentionally releasing a program or project from meeting a requirement
Will	used to indicate a statement of fact; not formally verified

A.2 ACRONYMS

Table A-2: Acronyms contains an alphabetized list of the definitions for abbreviations and acronyms used in this document.

Table A-2: Acronyms

Acronym	Definition
ACS	Aggregate Custody Signal
ADPCM	Adaptive Differential Pulse Code Modulation
AES	Advanced Encryption Standard
API	Application Programming Interface
ASIO	Asynchronous Input/Output
AVC	Advanced Video Coding
BCB	Block Confidentiality Block
BIBE	Bundle-In-Bundle Encapsulation
BIB	Block Integrity Block
BP	Bundle Protocol
BPCF	Block Processing Control Flags
BPGen	Bundle Protocol Generator
BPIng	Bundle Protocol Ingest
BPReceive	Bundle Protocol Receive
BPSec	Bundle Protocol Security
BPSend	Bundle Protocol Send
BPSink	Bundle Protocol Sink
BPSource	Bundle Protocol Source
BPv6	Bundle Protocol Version 6
BPv7	Bundle Protocol Version 7
CBHE	Compressed Bundle Header Encoding
CBOR	Concise Binary Object Representation
CCSDS	Consultative Committee for Space Data Systems
CGR	Contact Graph Routing
CLI	Command Line Interface
CM	Configuration Management
CMR	Contact Multigraph Routing
COSE	CBOR Object Signing and Encryption
CRC	Cyclic Redundancy Check
CSC	Computer Software Component
CSN	Checkpoint Serial Number
CSRC	Contributing Source
CTEB	Custody Transfer Enhancement Block
DD	Data Dictionary
DSA	Digital Signature Algorithm
DTN	Delay Tolerant Networking
EID	Endpoint Identifier
GCM	Galois/Counter Mode

Acronym	Definition
GRC	Glenn Research Center
GSM	Global System for Mobile Communication
GUI	Graphical User Interface
HEVC	High Efficiency Video Coding
HDTN	High-rate Delay Tolerant Networking
HMAC	Hash-based Message Authentication Code
I/O	Input/Output
IMA	Interactive Multimedia Association
IRTF	Internet Research Task Force
ITU	International Telecommunication Union
ITU-T	International Telecommunication Union Telecommunication Standardization Sector
JPEG	Joint Photographic Experts Group
KID	Key Identification
LPC	Linear Predictive Coding
LTP	Licklider Transmission Protocol
MPEG	Moving Picture Experts Group
MRU	Maximum Receive Unit
MTU	Maximum Transmission Unit
NASA	National Aeronautics and Space Administration
NPR	NASA Procedural Requirements
PARC	Palo Alto Research Center
PCM	Pulse Code Modulation
PDU	Protocol Data Unit
RFC	Request for Comment
RS	Report Segment
RTCP	Real-Time Transport Control Protocol
RTP	Real-Time Protocol
SCaN	Space Communications and Navigation
SDES	Stream Identifier Source Description
SDMP	Software Development and Management Plan
SDNV	Self-Delimiting Numeric Value
SEPG	Software Engineering Process Group
SHA	Secure Hash Algorithm
SHA2	Secure Hash Algorithm 2
SLIP	Serial Line Internet Protocol
SOMD	Space Operations Mission Directorate
SSL	Secure Socket Layer
SSRC	Synchronization Source Identifier
SWE	Software Engineering
SWEHB	Software Engineering Handbook
TBD	To Be Determined
TBR	To Be Resolved

Acronym	Definition
TCP	Transmission Control Protocol
TCPCL	Transmission Control Protocol Convergence Layer
TCPCLv3	Transmission Control Protocol Convergence Layer Version 3
TCPCLv4	Transmission Control Protocol Convergence Layer Version 4
TLS	Transport Layer Security
TLV	Type-Length-Value
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
UDPCL	User Datagram Protocol Convergence Layer
URI	Universal Resource Identifier
UTF	Unicode Transformation Format
UUID	Universally Unique Identifier
VVC	Versatile Video Coding

Appendix B TBD/TBR List

This appendix provides a list of all currently open TBD/TBR items contained within this document.

Where a TBD/TBR is included within the text of this document, they are incrementally numbered starting from DD001 (preceded by a "-", dash) and formatted in ***bold italics***. As an example: ***TBD-DD001*** for TBD or ***TBR-DD001*** for TBR.

Additionally, each TBD/TBR is entered into GitLab Issues in the HDTN SDF Repository as a task. Each task is assigned to a software team member responsible for addressing the need through completion.

Completion is defined here such that the TBD/TBR is addressed within this document and its respective task is closed out.

Table B-1: TBD/TBR List provides the list of TBDs/TBRs identified in this document. Each entry includes a short description, the task ID, status, and section of the document where the issue is located.

Table B-1: TBD/TBR List

TBD/TBR ID	Description	Task ID	Status	Section
TBD				
TBD-DD001	HDTN User Guide has not been released.	SDF #333	Open	2.0
TBR				
N/A	N/A	N/A	N/A	N/A

Appendix C Example: HDTN Run Script

The following is an example of a HDTN Run Script as described in HDTN-USER-035High-Rate Delay Tolerant Networking (HDTN) User Guide.

```
#!/bin/sh

# path variables
config_files=$HDTN_SOURCE_ROOT/config_files
hdtn_config=

$config_files/hdtn/hdtn_ingress1tcpclv4_port4556_egress1tcpclv4_port4558flowid2.json
sink_config=$config_files/inducts/bpsink_one_tcpclv4_port4558.json
gen_config=$config_files/outducts/bpgen_one_tcpclv4_port4556.json

cd $HDTN_SOURCE_ROOT

# bpsink
./build/common/bpcodec/apps/bpsink-async --my-uri-eid=ipn:2.1 --inducts-config-
file=$sink_config &
bpsink_PID=$!
sleep 3

# HDTN one process
# use the option --use-unix-timestamp when using a contact plan with unix timestamp
# use the option --use-mgr to use Multigraph Routing Algorithm (the default routing
Algorithm is CGR Dijkstra)
./build/module/hdtn_one_process/hdtn-one-process --hdtn-config-file=$hdtn_config --
contact-plan-file=contactPlanCutThroughMode.json &
oneprocess_PID=$!
sleep 10

#bpgen (configure bundle-rate=0 to send bundles at high rate)
./build/common/bpcodec/apps/bpgen-async --use-bp-version-7 --bundle-rate=100 --my-
uri-eid=ipn:1.1 --dest-uri-eid=ipn:2.1 --duration=40 --outducts-config-
file=$gen_config &
bpgen_PID=$!
sleep 8

# cleanup
sleep 30
echo "\nkilling bpgen1..." && kill -2 $bpgen_PID
sleep 2
echo "\nkilling egress..." && kill -2 $oneprocess_PID
sleep 2
echo "\nkilling bpsink2..." && kill -2 $bpsink_PID
```

Appendix D Example: BPSec Configuration File

The following is an example of a BPSec configuration file as documented in 12.1 Bundle Protocol Security Configuration.

```
{
    "bpsecConfigName": "my BPSec Config",
    "policyRules": [
        {
            "description": "Bpsource confidentiality",
            "securityPolicyRuleId": 1,
            "securityRole": "source",
            "securitySource": "ipn:1.1",
            "bundleSource": [
                "ipn:1.1"
            ],
            "bundleFinalDestination": [
                "ipn:2.1"
            ],
            "securityTargetBlockTypes": [
                1
            ],
            "securityService": "confidentiality",
            "securityContext": "aesGcm",
            "securityFailureEventSetReference": "default_confidentiality",
            "securityContextParams": [
                {
                    "paramName": "aesVariant",
                    "value": 256
                },
                {
                    "paramName": "ivSizeBytes",
                    "value": 12
                },
                {
                    "paramName": "keyFile",
                    "value": "config_files/bpsec/ipn1.1_confidentiality.key"
                },
                {
                    "paramName": "securityBlockCrc",
                    "value": 0
                },
                {
                    "paramName": "scopeFlags",
                    "value": 7
                }
            ]
        }
    ],
    "securityFailureEventSets": [
        {
            "name": "default_confidentiality",
            "description": "default bcb confidentiality security operations event
set",
            "securityOperationEvents": [
                {
```

```
                    "eventId": "sopCorruptedAtAcceptor",
                    "actions": [
                        "removeSecurityOperationTargetBlock"
                    ]
                }
            ]
        }
    ]
}
```

Appendix E Example: Contact Plan Configuration File

The following is an example of a contact plan configuration file as documented in 12.2 Contact Plan Configuration.

```
{
  "contacts": [
    {
      "contact": 0,
      "source": 32969,
      "dest": 32948,
      "startTime": 0,
      "endTime": 63072000,
      "rateBitsPerSec": 10000000000000
    },
    {
      "contact": 1,
      "source": 32948,
      "dest": 32969,
      "startTime": 0,
      "endTime": 63072000,
      "rateBitsPerSec": 10000000000000
    },
    {
      "contact": 2,
      "source": 32969,
      "dest": 32949,
      "startTime": 0,
      "endTime": 63072000,
      "rateBitsPerSec": 10000000000000
    },
    {
      "contact": 3,
      "source": 32949,
      "dest": 32969,
      "startTime": 0,
      "endTime": 63072000,
      "rateBitsPerSec": 10000000000000
    }
  ]
}
```

Appendix F Example: Distributed Configuration File

The following is an example of a distributed configuration file as documented in 12.3 Distributed Configuration.

```
{
    "zmqIngressAddress": "localhost",
    "zmqEgressAddress": "localhost",
    "zmqStorageAddress": "localhost",
    "zmqRouterAddress": "localhost",
    "zmqBoundIngressToConnectingEgressPortPath": 10100,
    "zmqConnectingEgressToBoundIngressPortPath": 10160,
    "zmqBoundEgressToConnectingRouterPortPath": 10162,
    "zmqConnectingEgressBundlesOnlyToBoundIngressPortPath": 10161,
    "zmqBoundIngressToConnectingStoragePortPath": 10110,
    "zmqConnectingStorageToBoundIngressPortPath": 10150,
    "zmqConnectingStorageToBoundEgressPortPath": 10120,
    "zmqConnectingStorageToBoundRouterPortPath": 10121,
    "zmqBoundEgressToConnectingStoragePortPath": 10130,
    "zmqConnectingRouterToBoundEgressPortPath": 10210,
    "zmqConnectingTelemToFromBoundIngressPortPath": 10301,
    "zmqConnectingTelemToFromBoundEgressPortPath": 10302,
    "zmqConnectingTelemToFromBoundStoragePortPath": 10303,
    "zmqConnectingTelemToFromBoundRouterPortPath": 10304
}
```

Appendix G Example: HDTN Configuration File

The following is an example of a HDTN configuration file as documented in 12.4 High-Rate Delay Tolerant Networking Configuration.

```
{
    "hdtnConfigName": "my hdtn config",
    "userInterfaceOn": true,
    "mySchemeName": "unused_scheme_name",
    "myNodeId": 1,
    "myBpEchoServiceId": 2047,
    "myCustodialSsp": "unused_custodial_ssp",
    "myCustodialServiceId": 0,
    "myRouterServiceId": 100,
    "isAcsAware": true,
    "acsMaxFillsPerAcsPacket": 100,
    "acsSendPeriodMilliseconds": 1000,
    "retransmitBundleAfterNoCustodySignalMilliseconds": 10000,
    "maxBundleSizeBytes": 10000000,
    "maxIngressBundleWaitOnEgressMilliseconds": 2000,
    "bufferRxToStorageOnLinkUpSaturation": false,
    "maxLtpReceiveUdpPacketSizeBytes": 65536,
    "neighborDepletedStorageDelaySeconds": 0,
    "fragmentBundlesLargerThanBytes": 0,
    "enforceBundlePriority": false,
    "zmqBoundRouterPubSubPortPath": 10200,
    "zmqBoundTelemApiPortPath": 10305,
    "inductsConfig": {
        "inductConfigName": "myconfig",
        "inductVector": [
            {
                "name": "stcp ingress",
                "convergenceLayer": "stcp",
                "boundPort": 4556,
                "numRxCircularBufferElements": 200,
                "keepAliveIntervalSeconds": 15
            }
        ]
    },
    "outductsConfig": {
        "outductConfigName": "myconfig",
        "outductVector": [
            {
                "name": "stcp egress1",
                "convergenceLayer": "stcp",
                "nextHopNodeId": 3,
                "remoteHostname": "node3",
                "remotePort": 4558,
                "maxNumberOfBundlesInPipeline": 500,
                "maxSumOfBundleBytesInPipeline": 50000000,
                "keepAliveIntervalSeconds": 17
            },
            {
                "name": "stcp egress2",
                "convergenceLayer": "stcp",
                "nextHopNodeId": 2,
```

```
                "remoteHostname": "node2",
                "remotePort": 4558,
                "maxNumberOfBundlesInPipeline": 500,
                "maxSumOfBundleBytesInPipeline": 50000000,
                "keepAliveIntervalSeconds": 17
            }
        ]
    },
    "storageConfig": {
        "storageImplementation": "asio_single_threaded",
        "tryToRestoreFromDisk": false,
        "autoDeleteFilesOnExit": true,
        "totalStorageCapacityBytes": 81920000000,
        "storageDeletionPolicy" : "never",
        "storageDiskConfigVector": [
            {
                "name": "d1",
                "storeFilePath": ".\/store1.bin"
            },
            {
                "name": "d2",
                "storeFilePath": ".\/store2.bin"
            }
        ]
    }
}
```