Workgroup: Delay-Tolerant Networking
Internet-Draft: draft-sipos-dtn-udpcl-01
Published: 25 March 2021
Intended Status: Standards Track
Expires: 26 September 2021
Authors: B. Sipos
RKF Engineering

Delay-Tolerant Networking UDP Convergence Layer Protocol

Abstract

This document describes a UDP-based convergence layer (UDPCL) for Delay-Tolerant Networking (DTN). This version of the UDPCL protocol clarifies requirements of RFC7122, adds discussion of multicast addressing, and updates to the Bundle Protocol (BP) contents, encodings, and convergence layer requirements in BP Version 7. Specifically, the UDPCL uses CBOR-encoded BPv7 bundles as its service data unit being transported and provides a reliable transport of such bundles. This version of UDPCL also includes security and extensibility mechanisms.

Status of This Memo

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

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

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

This Internet-Draft will expire on 26 September 2021.

Copyright Notice

Table of Contents

1. Introduction

This document describes the UDP-based convergence-layer protocol for Delay-Tolerant Networking. Delay-Tolerant Networking is an end-to-end architecture providing communications in and/or through highly stressed environments, including those with intermittent connectivity, long and/or variable delays, and high bit error rates. More detailed descriptions of the rationale and capabilities of these networks can be found in "Delay-Tolerant Network Architecture" [RFC4838].

An important goal of the DTN architecture is to accommodate a wide range of networking technologies and environments. The protocol used for DTN communications is the Bundle Protocol Version 7 (BPv7) [I-D.ietf-dtn-bpbis], an application-layer protocol that is used to construct a store-and-forward overlay network. BPv7 requires the services of a "convergence-layer adapter" (CLA) to send and receive bundles using the service of some "native" link, network, or Internet protocol. This document describes one such convergence-layer adapter that uses the well-known User Datagram Protocol (UDP). This convergence layer is referred to as UDP Convergence Layer (UDPCL). For the remainder of this document, the abbreviation "BP" without the version suffix refers to BPv7.

The locations of the UDPCL and the BP in the Internet model protocol stack (described in [RFC1122]) are shown in Figure 1. In particular, when BP is using UDP as its bearer with UDPCL as its convergence layer, both BP and UDPCL reside at the application layer of the Internet model.

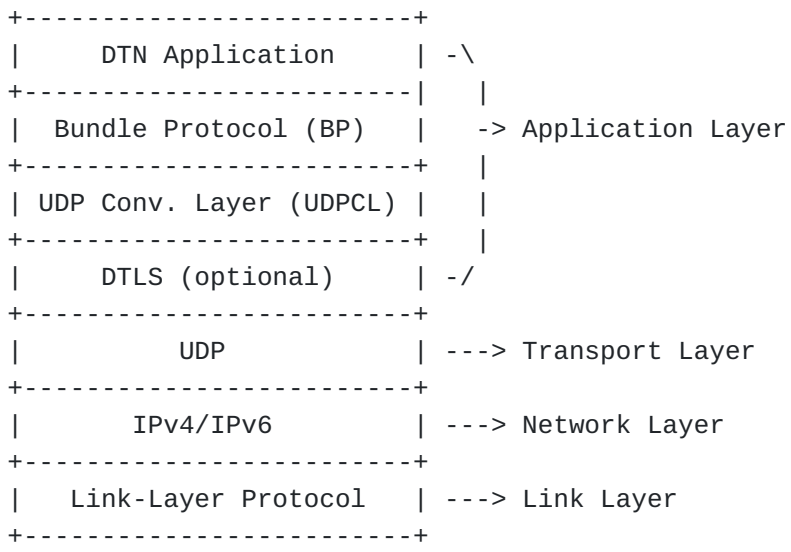

Figure 1: The Locations of the Bundle Protocol and the UDP Convergence-Layer Protocol above the Internet Protocol Stack

1.1. Scope

This document describes the format of the protocol data units passed between entities participating in UDPCL communications. This document does not address:

* The format of protocol data units of the Bundle Protocol, as those are defined elsewhere in [I-D.ietf-dtn-bpbis]. This includes the concept of bundle fragmentation or bundle encapsulation. The UDPCL transfers bundles as opaque data blocks.

* Mechanisms for locating or identifying other bundle entities (peers) within a network or across an internet. The mapping of Node ID to potential convergence layer (CL) protocol and network address is left to implementation and configuration of the BP Agent and its various potential routing strategies.

* Logic for routing bundles along a path toward a bundle's endpoint. This CL protocol is involved only in transporting bundles between adjacent entities in a routing sequence.

* Logic for performing rate control and congestion control of bundle transfers, both incoming and outgoing from a UDPCL entity.

* Policies or mechanisms for issuing Public Key Infrastructure Using X.509 (PKIX) certificates; provisioning, deploying, or accessing certificates and private keys; deploying or accessing certificate revocation lists (CRLs); or configuring security parameters on an individual entity or across a network.

* Uses of Datagram Transport Layer Security (DTLS) which are not based on PKIX certificate authentication (see Section 5.10.2) or in which authentication of both entities is not possible (see Section 5.10.1).

Any UDPCL implementation requires a BP agent to perform those above listed functions in order to perform end-to-end bundle delivery.

1.2. Use of CDDL

This document defines CBOR structure using the Concise Data Definition Language (CDDL) of [RFC8610]. The entire CDDL structure can be extracted from the XML version of this document using the XPath expression:

```
'//sourcecode[@type="cddl"]'
```

The following initial fragment defines the top-level symbols of this document's CDDL.

```
start = udpcl-ext-map
```

1.3. Requirements Language

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

1.4. Definitions Specific to the UDPCL Protocol

This section contains definitions specific to the UDPCL protocol.

UDPCL Entity: This is the notional UDPCL application that initiates UDPCL transfers. This design, implementation, configuration, and specific behavior of such an entity is outside of the scope of this document. However, the concept of an entity has utility within the scope of this document as the container and initiator of transfers. The relationship between a UDPCL entity and UDPCL sessions is defined as follows:

* A UDPCL Entity MAY actively perform any number of transfers and should do so whenever the entity has a bundle to forward to another entity in the network.

*A UDPCL Entity MAY support zero or more passive listening elements that listen for transfers from other entities in the network, including non-unicast transfers.

These relationships are illustrated in Figure 2. For the remainder of this document, the term "entity" without the prefix "UDPCL" refers to a UDPCL entity.

UDP Conversation: This refers to datagrams exchanged between two network peers, with each peer identified by a (unicast IP address, UDP port) tuple. Because UDP is connectionless, there is no notion of a conversation being "opened" or "closed" and some conversations are uni-directional.

Transfer: This refers to the procedures and mechanisms for conveyance of an individual bundle from one entity to one or more destinations. This version of UDPCL includes a fragmentation mechanism to allow transfers which are larger than the allowable UDP datagram size.

Transmit: This refers to a transfer outgoing from an entity as seen from that transmitting entity.

Receive: This refers to a transfer incoming to an entity as seen from that receiving entity.

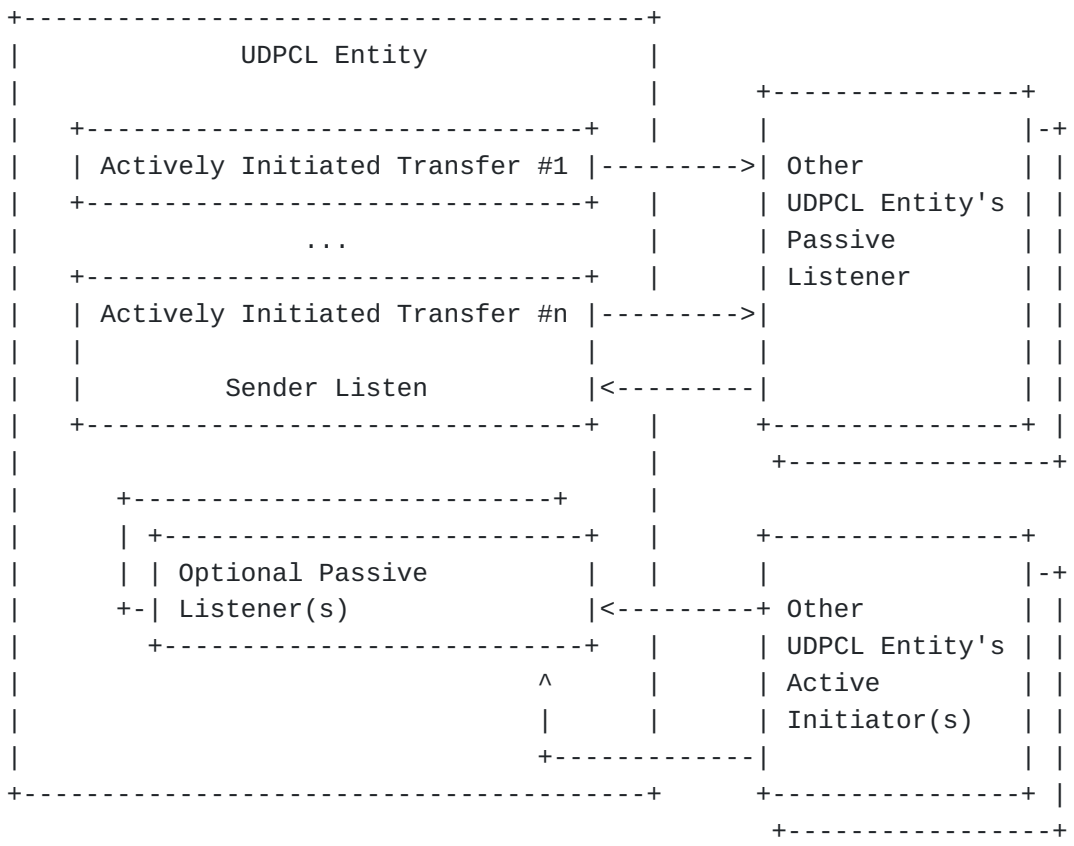

Figure 2: The relationships between UDPCL entities

2. General Protocol Description

The service of this protocol is the transmission of DTN bundles via the User Datagram Protocol (UDP). This document specifies the optional fragmentation of bundles, procedures for DTLS setup and teardown, and a set of messages and entity requirements. The general operation of the protocol is as follows.

Fundamentally, the UDPCL is a (logically) unidirectional "transmit and forget" protocol which itself maintains no long-term state and provides no feedback to the transmitter. The only long-term state related to UDPCL is used by DTLS in its session keeping (which is bound to a UDP conversation). An entity receiving a bundle from a particular source address-and-port does not imply that the transmitter is willing to accept bundle transfers on that same address-and-port. It is the obligation of a BP agent and its routing schemes to determine a bundle return path.

2.1. Convergence Layer Services

This version of the UDPCL provides the following services to support the over-laying Bundle Protocol agent. In all cases, this is not an API definition but a logical description of how the CL can interact

with the BP agent. Each of these interactions can be associated with any number of additional metadata items as necessary to support the operation of the CL or BP agent.

Begin Transmission: The principal purpose of the UDPCL is to allow a BP agent to transmit bundle data to one or more other entities. The receiver of each transfer is identified by an (destination) IPv4 or IPv6 address and a UDP port number (see Section 3 for details). The CL does not necessarily perform any transmission queueing, but may block while transmissions are being processed at the UDP layer. Any queueing of transmissions is the obligation of the BP agent.

Transmission Started: The UDPCL entity indicates to the BP agent when a bundle transmission begins sending UDP datagrams. Once started, there is no notion of a UDPCL transmission failure; a BP agent has to rely on bundle-level status reporting to track bundle progress through the network. Because of potential queueing or DTLS setup time, this may be delayed from the BP agent providing the bundle-to-transmit.

Transmission Finished: The UDPCL entity indicates to the BP agent when a bundle has been fully transmitted. This is not a positive indication that any next-hop receiver has either received or processed the transfer.

Reception Started: The UDPCL entity indicates to the BP agent when a bundle transfer has begun, which may include information about the total size of a fragmented transfer.

Reception Success: The UDPCL entity indicates to the BP agent when a bundle has been fully transferred from a peer entity. The transmitter of each transfer is identified by an (source) IP address and a UDP port number (see Section 3 for details).

Reception Failure: The UDPCL entity indicates to the BP agent on certain reasons for reception failure, notably upon an unfinished transfer timeout (see Section 3.5.2).

Attempt DTLS Session: The UDPCL allows a BP agent to preemptively attempt to establish a DTLS session with a peer entity (see Section 3.5.1 and Section 3.7). Each session attempt can send a

different set of session negotiation parameters as directed by the BP agent.

Close DTLS Session: The UDPCL allows a BP agent to preemptively close an established DTLS session with a peer entity. The closure request is on a per-session basis.

DTLS Session State Changed: The UDPCL entity indicates to the BP agent when a DTLS session state changes. The possible DTLS session states are defined in [RFC6347].

Begin Sender Listen: The UDPCL allows a BP agent to indicate when packets on a particular address-and-port is listened for (see Section 3.5.3). The Sender Listen interval is configurable for each peer address-and-port.

End Sender Listen: The UDPCL allows a BP agent to indicate when packets on a particular address-and-port are no longer be accepted.

Sender Listen Received: The UDPCL entity indicates to the BP agent when a Sender Listen extension has been received from a peer. The Sender Node ID, if present, is part of this indication.

2.2. PKIX Environments and CA Policy

This specification gives requirements about how to use PKIX certificates issued by a Certificate Authority (CA), but does not define any mechanisms for how those certificates come to be. The UDPCL uses the exact same mechanisms and makes the same assumptions as TCPCL in Section 3.4 of [I-D.ietf-dtn-tcpclv4].

2.3. Fragmentation Policies

It is a implementation matter for a sending entity to determine the path maximum transmit unit (PMTU) to be used as a target upper-bound UDP datagram size. Some techniques to perform MTU discovery are defined in [RFC8899]. All IP packets sent by a UDPCL entity SHOULD have the "don't fragment" bit set to allow detection of PMTU issues.

The priority order of fragmentation is the following:

1. When possible, bundles too large to fit in one PMTU-sized packet SHOULD be fragmented at the BP layer. Bundle payload fragmentation does not help a large bundle if extension blocks are a major contributor to bundle size, so in some circumstances BP layer fragmentation will not reduce the bundle size sufficiently. It is outside the scope of UDPCL to manage BP agent fragmentation policies; bundles are received from the BP agent either already fragmented or not.

2. Bundles too large to fit in one PMTU-sized packet SHALL be fragmented as a UDPCL transfer (see Section 3.6). Fragmentation at this level treats bundle transfers as opaque data, so it is independent of bundle block sizes or counts.

3. All IP packets larger than expected PMTU SHALL be fragmented by the transmitting entity to fit within one PMTU. Because of the issues listed in Section 3.2 of [RFC8085] and [RFC8900], it is best to avoid IP fragmentation as much as possible.

A UDPCL entity SHOULD NOT proactively drop an outgoing transfer due to datagram size. If intermediate network nodes drop IP packets it is an implementation matter to receive network feedback (e.g. ICMP Packet Too Big).

2.4. Error Checking Policies

The core Bundle Protocol specification assumes that bundles are transferring over an erasure channel, i.e., a channel that either delivers packets correctly or not at all.

A UDP transmitter SHALL NOT disable UDP checksums. A UDP receiver SHALL NOT disable the checking of received UDP checksums.

Even when UDP checksums are enabled, a small probability of UDP packet corruption remains. In some environments, it may be acceptable for a BP agent to occasionally receive corrupted input. In general, however, a UDPCL entity SHOULD insure the a bundle's blocks are either covered by a CRC or a BPSec integrity check.

2.5. Congestion Control Policies

The applications using UDPCL for bundle transport SHALL conform to the congestion control requirements of Section 3.1 of [RFC8085]. The application SHALL either perform active congestion control of bundles or behave as the Low Data-Volume application as defined in Section 3.1.3 of [RFC8085].

When nodes have bidirectional transfer capability, the bundle deletion reason code "traffic pared" can be used by a receiving agent to signal to the bundle source application that throttling of bundles along that path SHOULD occur.

3. UDPCL Operation

This section defines the UDPCL protocol and its interactions with under-layers (IP and UDP) and over-layers (BP), as illustrated in Figure 1. The section is organized from the network layer up toward the BP layer. It also discusses behavior within the UDPCL layer, which is illustrated in Figure 3.

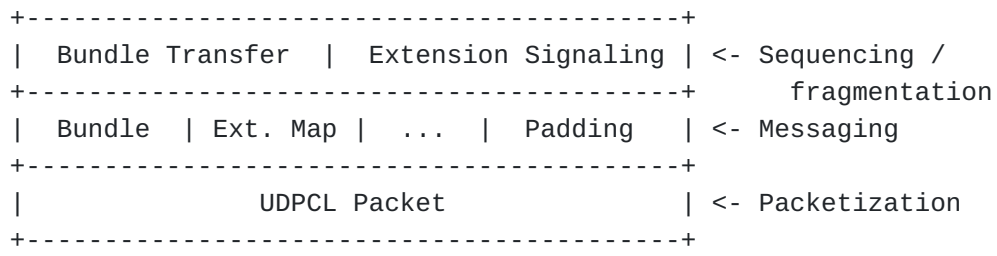

Figure 3: Breakdown of sub-layers within the UDPCL

3.1. IP Addressing

The earlier UDPCL specification in [RFC7122] did not include guidance on IP addressing, interface sourcing, or potential use of multicast, though the architecture of [RFC4838] explicitly includes multicast and anycast as expected network modes.

The BP agent determines the mapping from destination EID to next-hop CL parameters, including transfer destination address and transfer source interface. Some EIDs represent unicast destinations and others non-unicast destinations as defined in Section 4.2.5.1 of [I-D.ietf-dtn-bpbis]. The unicast-ness of an EID does not necessarily correspond with the unicast-ness, as some bundle routing schemes involve attempting multiple parallel paths to a unicast endpoint.

For unicast transfers to a single node, the destination address SHALL be a non-multicast IPv4 or IPv6 address (which does include link-local addresses). For unicast transfers, the source interface address MAY be supplied by the BP agent or otherwise determined by the operating system IP routing. When performing unicast transfers, a UDPCL entity SHOULD require DTLS use (see Section 3.7) or restrict the network to one protected by IPsec or some other under-layer security mechanism (e.g., a virtual private network).

For multicast transfers to one or more nodes, the destination address SHALL be a multicast IPv4 [IANA-IPv4-MCAST] or IPv6 [IANA-IPv6-MCAST] address. For multicast transfers, the source interface address MUST be supplied by the BP agent rather than inferred by the UDPCL entity.

3.2. UDP Header

Destination port number 4556 has been assigned by IANA [IANA-PORTS] as the Registered Port number for the UDP convergence layer and SHALL be used as a default. Other destination port numbers MAY be used per local configuration. Determining a passive entity's destination port number (if different from the registered UDPCL port number) is up to the implementation.

Any source port number MAY be used for UDPCL transfers. Typically an operating system assigned number in the UDP Ephemeral range (49152-65535) is used. For repeated messaging to the same destination address-and-port, the active entity SHOULD reuse the same source address-and-port. Reusing source address-and-port allows simplifies network monitoring and analysis and also enables bi-directional messaging as defined in Section 3.5.3.

3.3. UDPCL Packets

The lowest layer of UDPCL communication are individual-datagram packets. To exchange UDPCL data, an active entity SHALL transmit a UDP datagram to a listening passive entity in accordance with [RFC0768], typically by using the services provided by the operating system. For backward compatibility with [RFC7122], UDPCL has no explicit message type identifier.

Each UDP datagram SHALL contain one or more UDPCL message as defined in Section 3.4. Each type of message defines additional restrictions on how it may be used in a packet.

The following are special cases of UDPCL packet uses.

Unframed Transfer: An unframed transfer packet SHALL consist of a single encoded BPv6 or BPv7 bundle with no padding. This provides backward compatibility with [RFC7122] and a allows a trivial use of UDPCL which is just embedding an encoded bundle in a UDP datagram.

Keepalive A keepalive packet SHALL consist of exactly four octets of padding with no preceding message. This behavior maintains backward compatibility with [RFC7122].

3.4. UDPCL Messages

The middle layer of UDPCL communication are unframed, but self-delimited, messages. Specific message types MAY be concatenated together into a single packet, each message type indicates any restrictions on how it can be used within a packet.

For backward compatibility with [RFC7122], UDPCL has no explicit message type identifier. The message type is inferred by the inspecting the data contents according to the following rules:

BPv6 Bundle: All encoded BP version 6 bundles begin with the version identifier octet 0x06 in accordance with [RFC5050]. A message with a leading octet value of 0x06 SHALL be treated as a BPv6 bundle. Multiple BPv6 Bundles SHOULD NOT be present in one UDPCL packet to maintain compatibility with [RFC7122].

BPv7 Bundle:
All encoded BP version 7 bundles begin with a CBOR array head in accordance with [I-D.ietf-dtn-bpbis]. A message with a leading octet value indicating CBOR array (major type 4) SHALL be treated as a BPv7 bundle.

BPv7 bundles transmitted via UDPCL SHALL NOT include any leading CBOR tag. If the BP agent provides bundles with such tags the transmitting UDPCL entity SHALL remove them.

Extension Map: All UDPCL extensions SHALL be contained in a CBOR map in accordance with the definitions of Section 3.5. The encoded Extension Map SHALL NOT have any CBOR tags. A message with a leading octet value indicating CBOR map (major type 5) SHALL be treated as an Extension Map.

Padding: Padding data SHALL be a sequence of octets all with value 0x00. A message with a leading octet value of 0x00 SHALL be treated as padding.

Padding is used to ensure a UDP datagram is exactly a desired size. Because padding has no intrinsic length indication, if present it SHALL be the last contents of any UDPCL packet. A receiving UDPCL entity SHALL ignore all padding, including any trailing non-zero octets.

DTLS Record: In addition to the UDPCL specific messaging, immediately after a DTLS Initiation (see Section 3.5.1) the DTLS handshake sequence will begin. Data with a leading octet value of 0x16 SHALL be treated as a DTLS handshake record in accordance with Section 4.1 of [RFC6347].

If the datagram with the DTLS Initiation extension is not received by an entity, the entity SHOULD still detect the DTLS handshake records and start the handshake sequence at that point. Data with a leading octet value of 0x17--0x19 SHALL be treated as a DTLS sequencing failure; DTLS non-handshake records should never be seen by the UDPCL messaging layer.

A summary of how a receiving UDPCL entity can interpret the first octet of a datagram is listed in Table 1. When inspecting using CBOR major types, the range of values is caused by the CBOR head encoding of [RFC8949].

Octet Value	Message Content
0x00	Padding (remainder of packet)
0x06	BPv6 Bundle
0x16--0x19	DTLS Record (remainder of packet)

Octet Value	Message Content
0x80--0x9F	BPv7 Bundle (CBOR array)
0xA0--0xBF	Extension Map (CBOR map)
others	unused

Table 1: First-Octet Contents

3.5. UDPCL Extension Items

Extensions to UDPCL are encoded per-datagram in a single CBOR map as defined in Section 3.4. Each UDPCL extension item SHALL be identified by a unique Extension ID used as a key in the Extension Map. Extension ID values SHALL be a CBOR int item no longer than 16-bits. Extension ID assignments are listed in Section 6.2.

Unless prohibited by particular extension type requirements, a single Extension Map MAY contain any combination of extension items. Receivers SHALL ignore extension items with unknown Extension ID and continue to process known extension items.

```
; Map structure requiring non-zero-int keys.
; CDDL cannot enforce type-specific requirements about other items
; being present (or not present) in the same map.
udpcl-ext-map = $udpcl-ext-map .within udpcl-ext-map-structure
$udpcl-ext-map = {
    * $$udpcl-ext-item
}
udpcl-ext-map-structure = {
    * ext-key => any
}
ext-key = (int .size 2) .ne 0
```

Figure 4: Extension Map structure CDDL

The following subsections define the initial UDPCL extension types.

3.5.1. DTLS Initiation (STARTTLS)

This extension item indicates that the transmitter is about to begin a DTLS handshake sequence in accordance with Section 3.7.

The DTLS Initiation value SHALL be an untagged null value. There are no DTLS parameters actually transmitted as part of this extension, it only serves to indicate to the recipient that the next datagram will be a DTLS ClientHello. Although the datagram containing this extension is not retransmitted, the DTLS handshake itself will retransmit ClientHello messages until confirmation is received.

```
$$udpcl-ext-item //= (
    5: null
)
```

Figure 5: DTLS Initiation CDDL

If the entity is configured to enable exchanging messages according to DTLS 1.2 [RFC6347] or any successors which are compatible with that DTLS ClientHello, the first message in any sequence to a unicast recipient SHALL be an Extension Map with the DTLS Initiation item. The RECOMMENDED policy is to enable DTLS for all unicast recipients, even if security policy does not allow or require authentication. This follows the opportunistic security model of [RFC7435], though an active attacker could interfere with the exchange in such cases (see Section 5.3).

The Extension Map containing a DTLS Initiation item SHALL NOT contain any other items. A DTLS Initiation item SHALL NOT be present in any message transmitted within a DTLS session. A receiver of a DTLS Initiation item within a DTLS session SHALL ignore it. Between transmitting a DTLS Initiation item and finishing a DTLS handshake (either success or failure) an entity SHALL NOT transmit any other UDP datagrams in that same conversation.

3.5.2. Bundle Transfer

This extension item allows CL-layer fragmentation of bundle transfers as defined in Section 3.6.

The Transfer value SHALL be an untagged CBOR array of four items. The items are defined in the following order:

Transfer ID: This field SHALL be a CBOR uint item no larger than 32-bits, which is used to correlate multiple fragments.

Total Length: This field SHALL be a CBOR uint item no larger than 32-bits, which is used to indicate the total length (in octets) of the transfer. If multiple Transfer items for the same Transfer ID are received with differing Total Length values, the receiver SHALL treat the transfer as being malformed and refuse to handle any further fragments associated with the transfer.

Fragment Offset: This field SHALL be a CBOR uint item no larger than 32-bits, which is used to indicate the offset (in octets) into the transfer for the start of this fragment.

Fragment Data: This field SHALL be a CBOR bstr item no larger than 2^32-1 octets, in which the fragment data is contained. The bstr itself indicates the length of the fragment data.

```
$$udpcl-ext-item //= (
    2: [
        transfer-id: uint .size 4,
        total-length: uint .size 4,
        fragment-offset: uint .size 4,
        fragment-data: bstr,
    ]
)
```

Figure 6: Transfer CDDL

3.5.3. Sender Listen

This extension item indicates that the transmitter is listening for UDPCL packets on the source address-and-port used to transmit the message containing this extension item. This is different from simply listening on a UDP port (either the default or any other) when the entity is behind a NAT or firewall which will not allow unsolicited UDP/IP datagrams. Although the packet containing this extension is not retransmitted, the time interval is finite and the extension is sent repeatedly while the transmitter continues to listen for packets. There is no positive indication that packets are no longer accepted; the Sender Listen just stops being transmitted.

The Sender Listen value SHALL be an untagged uint value representing the interval of time (in milliseconds) that the entity is willing to accept UDPCL packets on the source address-and-port used for the associated transmitted message. After transmitting a Sender Listen, the entity SHALL listen for and accept datagrams on the source address-and-port used for the associated transmitted message. As long as the entity is still willing to accept packets, at the end of one accept interval the entity SHALL transmit another Sender Listen item. This repetition continues until the entity is no longer willing to listen for packets.

A receiving entity SHOULD treat a peer as no longer listening after an implementation-defined timeout since the last received Sender Listen item. A RECOMMENDED Sender Listen timeout is three (3) times the associated time duration; this allows a single dropped datagram to not interrupt a continuous sequence.

```
$$udpcl-ext-item //= (
    3: time-duration,
)
time-duration = uint
```

Figure 7: Sender Listen CDDL

Unlike the generic source port requirement in Section 3.2, when repeated Sender Listen are transmitted in a sequence a consistent source address-and-port SHALL be used.

The Sender Listen interval SHOULD be no shorter than 1 second and no longer than 60 seconds.

An entity SHOULD include a Sender Node ID item along with a Sender Listen item if the conditions of Section 3.5.4 are met. An entity MAY include any other extension type along with a Sender Listen item. An entity SHALL NOT transmit a Sender Listen item before or along with a DTLS Initiate if DTLS is desired for a conversation.

This extension is not a neighbor discovery mechanism and does not indicate an entity listening generally on a particular UDP port. Sender Listen applies only to UDP datagrams from the the peer address-and-port. An entity SHALL NOT include a Sender Listen item in a message transmitted to a multicast address.

3.5.4. Sender Node ID

This extension item indicates the Node ID of the transmitter. For DTLS-secured sessions (see Section 3.7.4) this extension can be used to disambiguate an end-entity certificate which has multiple NODE-ID values.

The Sender Node ID value SHALL be an untagged tstr value containing a Node ID. Every Node ID SHALL be a URI consistent with the requirements of [RFC3986] and the URI schemes of the IANA "Bundle Protocol URI Scheme Type" registry [IANA-BUNDLE].

```
$$udpcl-ext-item //= (
    4: nodeid,
)
nodeid = tstr
```

Figure 8: Sender Node ID CDDL

An entity SHOULD NOT include a Sender Node ID item if a DTLS session has already been established and the presented end-entity certificate contains a single NODE-ID. In this case there is no ambiguity about which Node ID is identified by the certificate.

If an entity receives a peer Node ID which is not authenticated (by the procedure of Section 3.7.4) that Node ID SHOULD NOT be used by a BP agent for any discovery or routing functions. Trusting an unauthenticated Node ID can lead to the threat described in Section 5.7.

3.6. Explicit Transfers

This version of UDPCL supports CL-layer fragmentation of bundles larger than the PMTU would otherwise allow. Policies related to fragmentation at, above, or below the UDPCL layer are defined in Section 2.3. The entire fragmented bundle is referred to as a Transfer and individual fragments of a transfer are encoded as Transfer extension items in accordance with Section 3.5.2.

This mechanism also allows a bundle transfer to be transmitted along with additional extension items, which the unframed bundle-in-datagram data does not. This specification does not define any extension items which augment an associated transfer.

3.6.1. Bundle Transfer ID

Each Transfer item contains a Transfer ID which is used to correlate messages for a single bundle transfer. A Transfer ID does not attempt to address uniqueness of the bundle data itself and has no relation to concepts such as bundle fragmentation. Each invocation of UDPCL by the BP agent, requesting transmission of a bundle (fragmentary or otherwise), can cause the initiation of a single UDPCL transfer.

Because UDPCL operation is connectionless, Transfer IDs from each entity SHALL be unique for the operating duration of the entity. In practice, the ID needs only be unique for the longest receiver reassembly time window; but because that information is not part of the protocol there is no way for an transmitting entity to know the reassembly time window of any receiver (see Section 3.6.2). When there are bidirectional bundle transfers between UDPCL entities, an entity SHOULD NOT rely on any relation between Transfer IDs originating from each side of the conversation.

Although there is not a strict requirement for Transfer ID initial values or ordering (see Section 5.11), in the absence of any other mechanism for generating Transfer IDs an entity SHALL use the following algorithm: the initial Transfer ID from each entity is zero; subsequent Transfer ID values are incremented from the prior Transfer ID value by one; upon exhaustion of the entire 32-bit Transfer ID space, the subsequent Transfer ID value is zero.

3.6.2. Fragmentation and Reassembly

The full data content of a transfer SHALL be an unframed (BPv6 or BPv7) bundle as defined in Section 3.4. A receiving entity SHALL discard any reassembled transfer which does not properly contain a bundle.

A transmitting entity MAY produce a Transfer with a single fragment (i.e., a Fragment Data size identical to the Total Length). A transmitting entity SHALL NOT produce Transfer fragments with overlapping span. A transmitting entity SHOULD transmit Transfer fragments in order of Fragment Offset; this makes the behavior deterministic.

Because of the nature of UDP transport, there is no guaranteed order or timing of received Transfer items. A receiving entity SHALL consider a transport as finished when Fragment Data has been received which fully covers the Total Length of the transfer.

A receiving entity SHALL discard any Transfer item containing different CBOR types than defined in this document. A receiving entity SHALL discard any Transfer item containing a fragment with an overlapping span. Because there is no feedback indication at the UDPCL layer, a transmitter has no indication when a transfer is discarded by the receiver.

A receiving entity SHOULD discard unfinished transfer state after an implementation-defined timeout since the last received fragment. Entities SHOULD choose a transfer timeout interval no longer than one minute (60 seconds). Discarding an unfinished transfer causes no indication to the transmitting entity, but does indicate this to the BP agent. This timeout is purely receiver-side and represents the maximum allowed time between sequential received datagrams (in any order), which should be short if the datagrams take a similar network path.

3.7. UDPCL Security

This version of the UDPCL supports establishing a DTLS session within an existing UDP conversation. When DTLS is used within the UDPCL it affects the entire conversation. There is no concept of a plaintext message being sent in a conversation after a DTLS session is established.

Once established, the lifetime of a DTLS session SHALL be bound by the DTLS session ticket lifetime or either peer sending a Closure Alert record.

Subsequent DTLS session attempts to the same passive entity MAY attempt to use the DTLS session resumption feature. There is no guarantee that the passive entity will accept the request to resume a DTLS session, and the active entity cannot assume any resumption outcome.

3.7.1. Entity Identification

The UDPCL uses DTLS for certificate exchange in both directions to identify each entity and to allow each entity to authenticate its peer. Each certificate can potentially identify multiple entities and there is no problem using such a certificate as long as the identifiers are sufficient to meet authentication policy (as described in later sections) for the entity which presents it.

The types and priorities of identities used by DTLS in UDPCL is the same as those for TLS in TCPCL as defined in Section 4.4.1 of [I-D.ietf-dtn-tcpclv4].

3.7.2. Certificate Profile for UDPCL

All end-entity certificates used by a UDPCL entity SHALL conform to the profile defined in Section 4.4.2 of [I-D.ietf-dtn-tcpclv4].

3.7.3. DTLS Handshake

The signaling for DTLS Initiation is described in Section 3.5.1. After sending or receiving an Extension Map containing a DTLS Initiation item, an entity SHALL begin the handshake procedure of Section 4.2 of [RFC6347]. By convention, this protocol uses the entity which sent the DTLS Initiation (the active peer) as the "client" role of the DTLS handshake request.

Upon receiving an unexpected ClientHello record outside of a DTLS session, an entity SHALL begin the DTLS handshake procedure as if a DTLS Initiation had been received. This allows recovering from a dropped packet containing DTLS Initiation.

3.7.4. DTLS Authentication

The function and mechanism of DTLS authentication in UDPCL is the same as for TLS in TCPCL as defined in Section 4.4.4 of [I-D.ietf-dtn-tcpclv4], with the exception that Node ID Authentication is based on an optional Sender Node ID extension (see Section 3.5.4) used to disambiguate when an end-entity certificate contains multiple NODE-ID values.

3.7.5. Policy Recommendations

The policy recommendations given here are are the same as those for TCPCL in Section 4.4.5 of [I-D.ietf-dtn-tcpclv4]. They are restated in this document for clarity.

A RECOMMENDED security policy is to enable the use of OCSP checking during DTLS handshake. A RECOMMENDED security policy is that if an Extended Key Usage is present that it needs to contain "id-kp-

bundleSecurity" of [IANA-SMI] to be usable with UDPCL security. A RECOMMENDED security policy is to require a validated NODE-ID and to ignore any network-level DNS-ID or IPADDR-ID.

This policy relies on and informs the certificate requirements in Section 3.7.2. This policy assumes that a DTN-aware CA (see Section 2.2) will only issue a certificate for a Node ID when it has verified that the private key holder actually controls the DTN node; this is needed to avoid the threat identified in Section 5.7. This policy requires that a certificate contain a NODE-ID and allows the certificate to also contain network-level identifiers. A tailored policy on a more controlled network could relax the requirement on Node ID validation and allow just network-level identifiers to authenticate a peer.

3.7.6. Example Secured and Bidirectional Transfers

This simple example shows a sequence of pre-transfer setup followed by a set of (unrelated) bundle transfers. All messaging in this example occurs between the same Entity A address-and-port and Entity B address-and-port.

The example Entity A has a policy to only send or receive bundles within a DTLS session, so any outgoing bundles to Entity B are queued until the DTLS session is established. Because Entity A is willing to accept transfers on its ephemeral UDP port, the first outgoing message after the DTLS handshake contains the Sender Listen extension (along with a Sender Node ID indicating its identity to Entity B).

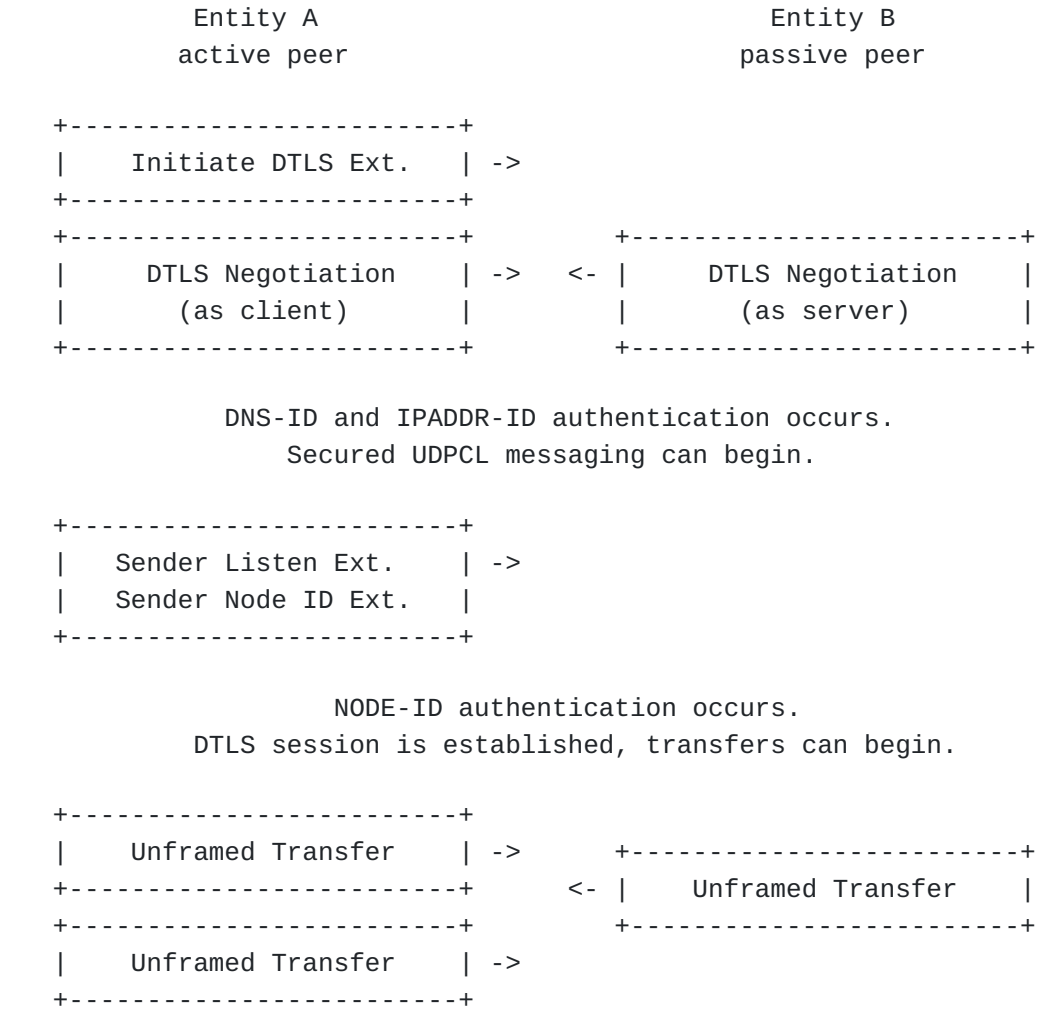

Figure 9: An example of the flow of protocol messages on a single UDP conversation between two entities

4. Implementation Status

This section is to be removed before publishing as an RFC.

[NOTE to the RFC Editor: please remove this section before publication, as well as the reference to [RFC7942], [github-dtn-demo-agent], and [github-dtn-wireshark].]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available

implementations or their features. Readers are advised to note that other implementations can exist.

An example implementation of the this draft of UDPCL has been created as a GitHub project [github-dtn-demo-agent] and is intended to use as a proof-of-concept and as a possible source of interoperability testing. This example implementation uses D-Bus as the CL-BP Agent interface, so it only runs on hosts which provide the Python "dbus" library.

A wireshark dissector for UDPCL has been created as a GitHub project [github-dtn-wireshark] and has been kept in-sync with the latest encoding of this specification.

5. Security Considerations

This section separates security considerations into threat categories based on guidance of BCP 72 [RFC3552].

5.1. Threat: Passive Leak of Node Data

When used without DTLS security, the UDPCL can expose the Node ID and other configuration data to passive eavesdroppers. This can occur even if no bundle transfers are transmitted. This can be avoided by always using DTLS, even if authentication is not available (see Section 5.10).

5.2. Threat: Passive Leak of Bundle Data

UDPCL can be used to provide point-to-point unicast transport security, but does not provide multicast security, security of data-at-rest, and does not guarantee end-to-end bundle security. In those cases the bundle security mechanisms defined in [I-D.ietf-dtn-bpsec] are to be used instead.

When used without DTLS security, the UDPCL exposes all bundle data to passive eavesdroppers. This can be avoided by always using DTLS for unicast messaging, even if authentication is not available (see Section 5.10).

5.3. Threat: Transport Security Stripping

When security policy allows non-DTLS messaging, UDPCL does not protect against active network attackers. It is possible for a on-path attacker to drop or alter packets containing Extension Map and/or DTLS handshake records, which will cause the receiver to not negotiate a DTLS session. This leads to the "SSL Stripping" attack described in [RFC7457].

When DTLS is available on an entity, it is strongly encouraged that the security policy disallow non-DTLS messaging for unicast purposes. This requires that the DTLS handshake occurs before any other UDPCL messaging, regardless of the policy-driven parameters of the handshake and policy-driven handling of the handshake outcome.

One mechanism to mitigate the possibility of DTLS stripping is the use of DNS-based Authentication of Named Entities (DANE) [RFC6698] toward the passive peer. This mechanism relies on DNS and is unidirectional, so it doesn't help with applying policy toward the active peer, but it can be useful in an environment using opportunistic security. The configuration and use of DANE are outside of the scope of this document.

The negotiated use of DTLS is identical behavior to STARTTLS use in [RFC2595], [RFC4511], and others.

5.4. Threat: Weak DTLS Configurations

Even when using DTLS to secure the UDPCL session, the actual ciphersuite negotiated between the DTLS peers can be insecure. Recommendations for ciphersuite use are included in BCP 195 [RFC7525]. It is up to security policies within each UDPCL entity to ensure that the negotiated DTLS ciphersuite meets transport security requirements.

5.5. Threat: Untrusted End-Entity Certificate

The profile in Section 3.7.4 uses end-entity certificates chained up to a trusted root CA. During DTLS handshake, either entity can send a certificate set which does not contain the full chain, possibly excluding intermediate or root CAs. In an environment where peers are known to already contain needed root and intermediate CAs there is no need to include those CAs, but this has a risk of an entity not actually having one of the needed CAs.

5.6. Threat: Certificate Validation Vulnerabilities

Even when DTLS itself is operating properly an attacker can attempt to exploit vulnerabilities within certificate check algorithms or configuration to establish a secure DTLS session using an invalid certificate. An invalid certificate exploit could lead to bundle data leaking and/or denial of service to the Node ID being impersonated.

There are many reasons, described in [RFC5280] and [RFC6125], why a certificate can fail to validate, including using the certificate outside of its valid time interval, using purposes for which it was not authorized, or using it after it has been revoked by its CA. Validating a certificate is a complex task and can require network

connectivity outside of the primary UDPCL network path(s) if a mechanism such as OCSP [RFC6960] is used by the CA. The configuration and use of particular certificate validation methods are outside of the scope of this document.

5.7. Threat: BP Node Impersonation

The certificates exchanged by DTLS enable authentication of peer DNS name and Node ID, but it is possible that a peer either not provide a valid certificate or that the certificate does not validate either the DNS-ID/IPADDR-ID or NODE-ID of the peer (see Section 2.2). Having a CA-validated certificate does not alone guarantee the identity of the network host or BP node from which the certificate is provided; additional validation procedures in Section 3.7.3 bind the DNS-ID/IPADDR-ID or NODE-ID based on the contents of the certificate.

The DNS-ID/IPADDR-ID validation is a weaker form of authentication, because even if a peer is operating on an authenticated network DNS name or IP address it can provide an invalid Node ID and cause bundles to be "leaked" to an invalid node. Especially in DTN environments, network names and addresses of nodes can be time-variable so binding a certificate to a Node ID is a more stable identity.

NODE-ID validation ensures that the peer to which a bundle is transferred is in fact the node which the BP Agent expects it to be. In circumstances where certificates can only be issued to DNS names, Node ID validation is not possible but it could be reasonable to assume that a trusted host is not going to present an invalid Node ID. Determining when a DNS-ID/IPADDR-ID authentication can be trusted to validate a Node ID is also a policy matter outside of the scope of this document.

One mitigation to arbitrary entities with valid PKIX certificates impersonating arbitrary Node IDs is the use of the PKIX Extended Key Usage key purpose "id-kp-bundleSecurity" of [IANA-SMI]. When this Extended Key Usage is present in the certificate, it represents a stronger assertion that the private key holder should in fact be trusted to operate as a DTN Node.

5.8. Threat: Denial of Service

The behaviors described in this section all amount to a potential denial-of-service to a UDPCL entity. The denial-of-service could be limited to an individual UDPCL entity, or could affect all entities on a host or network segment.

An entity can send a large amount of data to a UDPCL entity, requiring the receiving entity to handle the data. The victim entity

can block UDP packets from network peers which are thought to be incorrectly behaving within network.

An entity can also send only one fragment of a seemingly valid transfer and never send the remaining fragments, which will cause resources on the receiver to be wasted on transfer reassembly state. The victim entity can either block packets from network peers or intentionally keep a short unfinished transfer timeout (see Section 3.6.2).

The keepalive mechanism can be abused to waste throughput within a network link which would otherwise be usable for bundle transmissions.

5.9. Mandatory-to-Implement DTLS

Following IETF best current practice, DTLS is mandatory to implement for all UDPCL implementations but DTLS is optional to use for a any given transfer. The recommended configuration of Section 3.5.1 is to always attempt DTLS, but entities are permitted to disable DTLS based on local configuration. The configuration to enable or disable DTLS for an entity or a session is outside of the scope of this document. The configuration to disable DTLS is different from the threat of DTLS stripping described in Section 5.3.

5.10. Alternate Uses of DTLS

This specification makes use of PKIX certificate validation and authentication within DTLS. There are alternate uses of DTLS which are not necessarily incompatible with the security goals of this specification, but are outside of the scope of this document. The following subsections give examples of alternate DTLS uses.

5.10.1. DTLS Without Authentication

In environments where PKI is available but there are restrictions on the issuance of certificates (including the contents of certificates), it may be possible to make use of DTLS in a way which authenticates only the passive entity of a UDPCL transfer or which does not authenticate either entity. Using DTLS in a way which does not successfully authenticate some claim of both peer entities of a UDPCL transfer is outside of the scope of this document but does have similar properties to the opportunistic security model of [RFC7435].

5.10.2. Non-Certificate DTLS Use

In environments where PKI is unavailable, alternate uses of DTLS which do not require certificates such as pre-shared key (PSK) authentication [RFC5489] and the use of raw public keys [RFC7250]

are available and can be used to ensure confidentiality within UDPCL. Using non-PKI node authentication methods is outside of the scope of this document.

5.11. Predictability of Transfer IDs

The only requirement on Transfer IDs is that they are unique from the transmitting peer only. The trivial algorithm of the first transfer starting at zero and later transfers incrementing by one causes absolutely predictable Transfer IDs. Even when UDPCL is not DTLS secured and there is a on-path attacker altering UDPCL messages, there is no UDPCL feedback mechanism to interrupt or refuse a transfer so there is no benefit in having unpredictable Transfer IDs.

6. IANA Considerations

Registration procedures referred to in this section are defined in [RFC8126].

6.1. Port Number

Within the port registry of [IANA-PORTS], UDP port number 4556 has been previously assigned as the default port for the UDP convergence layer in [RFC7122]. This assignment to UDPCL is unchanged, but the assignment reference is updated to this specification. There is no UDPCL version indication on-the-wire but this specification is a superset of [RFC7122] and is fully backward compatible. The related assignment for DCCP port 4556 (registered by [RFC7122]) is unchanged.

Parameter	Value
Service Name:	dtn-bundle
Transport Protocol(s):	UDP
Assignee:	IESG <iesg@ietf.org>
Contact:	IESG <iesg@ietf.org>
Description:	DTN Bundle UDP CL Protocol
Reference:	This specification.
Port Number:	4556

Table 2

6.2. UDPCL Extension Types

EDITOR NOTE: sub-registry to-be-created upon publication of this specification.

IANA will create, under the "Bundle Protocol" registry [IANA-BUNDLE], a sub-registry titled "Bundle Protocol UDP Convergence-Layer Extension Types" and initialize it with the contents of Table

3. For positive code points the registration procedure is Specification Required. Negative code points are reserved for use on private networks for functions not published to the IANA.

Specifications of new extension types need to define the CBOR item structure of the extension data as well as the purpose and relationship of the new extension to existing session/transfer state within the baseline UDPCL sequencing. Receiving entities will ignore items with unknown Extension ID, and that behavior needs to be considered by new extension types.

Expert(s) are encouraged to be biased towards approving registrations unless they are abusive, frivolous, or actively harmful (not merely aesthetically displeasing, or architecturally dubious).

Extension ID	Name	References
negative	Private/Experimental Use	This specification.
0	Reserved	This specification.
2	Transfer	Section 3.5.2 of this specification.
3	Sender Listen	Section 3.5.3 of this specification.
4	Sender Node ID	Section 3.5.4 of this specification.
5	DTLS Initiation (STARTTLS)	Section 3.5.1 of this specification.
6-65535	Unassigned	

Table 3: Extension Type Codes

7. Acknowledgments

TBD

8. References

8.1. Normative References

[IANA-BUNDLE] IANA, "Bundle Protocol", <https://www.iana.org/assignments/bundle/>.

[IANA-PORTS] IANA, "Service Name and Transport Protocol Port Number Registry", <https://www.iana.org/assignments/service-names-port-numbers/>.

[IANA-IPv4-MCAST] IANA, "IPv4 Multicast Address Space Registry", <https://www.iana.org/assignments/multicast-addresses/>.

[IANA-IPv6-MCAST] IANA, "IPv6 Multicast Address Space Registry", <https://www.iana.org/assignments/ipv6-multicast-addresses/>.

[IANA-SMI] IANA, "Structure of Management Information (SMI) Numbers", <https://www.iana.org/assignments/smi-numbers/>.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5050] Scott, K. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, DOI 10.17487/RFC5050, November 2007, <https://www.rfc-editor.org/info/rfc5050>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/info/rfc6960>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/

RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[I-D.ietf-dtn-bpbis]
Burleigh, S., Fall, K., and E. Birrane, "Bundle Protocol Version 7", Work in Progress, Internet-Draft, draft-ietf-dtn-bpbis-31, 25 January 2021, <https://tools.ietf.org/html/draft-ietf-dtn-bpbis-31>.

[I-D.ietf-dtn-tcpclv4]
Sipos, B., Demmer, M., Ott, J., and S. Perreault, "Delay-Tolerant Networking TCP Convergence Layer Protocol Version 4", Work in Progress, Internet-Draft, draft-ietf-dtn-tcpclv4-24, 7 December 2020, <https://tools.ietf.org/html/draft-ietf-dtn-tcpclv4-24>.

8.2. Informative References

[RFC2595] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, DOI 10.17487/RFC2595, June 1999, <https://www.rfc-editor.org/info/rfc2595>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[RFC4511] Sermersheim, J., Ed., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, DOI 10.17487/RFC4511, June 2006, <https://www.rfc-editor.org/info/rfc4511>.

[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <https://www.rfc-editor.org/info/rfc4838>.

[RFC5489] Badra, M. and I. Hajjeh, "ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)", RFC 5489, DOI 10.17487/RFC5489, March 2009, <https://www.rfc-editor.org/info/rfc5489>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/info/rfc6698>.

[RFC7122] Kruse, H., Jero, S., and S. Ostermann, "Datagram Convergence Layers for the Delay- and Disruption-Tolerant

Networking (DTN) Bundle Protocol and Licklider Transmission Protocol (LTP)", RFC 7122, DOI 10.17487/RFC7122, March 2014, <https://www.rfc-editor.org/info/rfc7122>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <https://www.rfc-editor.org/info/rfc7435>.

[RFC7457] Sheffer, Y., Holz, R., and P. Saint-Andre, "Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)", RFC 7457, DOI 10.17487/RFC7457, February 2015, <https://www.rfc-editor.org/info/rfc7457>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC8899] Fairhurst, G., Jones, T., Tüxen, M., Rüngeler, I., and T. Völker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <https://www.rfc-editor.org/info/rfc8899>.

[RFC8900] Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <https://www.rfc-editor.org/info/rfc8900>.

[I-D.ietf-dtn-bpsec] Birrane, E. and K. McKeever, "Bundle Protocol Security Specification", Work in Progress, Internet-Draft, draft-ietf-dtn-bpsec-26, 8 January 2021, <https://tools.ietf.org/html/draft-ietf-dtn-bpsec-26>.

[github-dtn-demo-agent] Sipos, B., "UDPCL Example Implementation", <https://github.com/BSipos-RKF/dtn-demo-agent/>.

[github-dtn-wireshark] Sipos, B., "UDPCL Wireshark Dissector", <https://github.com/BSipos-RKF/dtn-wireshark/>.

Appendix A. Significant changes from RFC7122

The areas in which changes from [RFC7122] have been made to existing requirements:

- *Made explicit references to UDP- and IP-related RFCs.
- *Made more strict Keepalive and Padding requirements.
- *Defined UDPCL security and made mandatory-to-implement.

The areas in which extensions from [RFC7122] have been made as new behaviors are:

- *Added BPv7 bundle as a possible UDPCL payload.
- *Added Extension Map message type and initial extension types.
- *Defined semantics for UDPCL multicast addressing.

Author's Address

Brian Sipos
RKF Engineering Solutions, LLC
7500 Old Georgetown Road
Suite 1275
Bethesda, MD 20814-6198
United States of America

Email: BSipos@rkf-eng.com