JOHNS HOPKINS
APPLIED PHYSICS LABORATORY

Bundle Protocol Version 7 Administrative Record Types Registry

IETF 113 DTN WG

Brian Sipos
JHU/APL

Need For this Updating Document

- RFC9171 defines an explicit table of Administrative Record Types code points.
- There is a pre-existing IANA sub-registry of BPv6 Admin. Record Types.
  - The IANA table was missing the CCSDS Aggregate Custody Signal allocation.
- Other pre-existing IANA sub-registries with BPv6-7 overlap were updated to include a “Bundle Protocol Version” column, which disambiguates and allows for overlapping registrations.
- This proposed document updates the Admin. Record Types sub-registry to be similar to the others with BPv6-7 overlap.
  - It makes an explicit reservation of code point zero.
  - It adds a high-valued reservation for private or experimental use in the 32-bit-encoded range. This leaves the full 16-bit space available for BPv7 use.
- No change is made to the “Specification Required” registration procedure.

What the Changes Look Like

Bundle Protocol Version	Value	Description	Reference
6,7	0	Reserved	[RFC7116] [This specification]
6,7	1	Bundle status report	[RFC5050] [RFC9171]
6	2	Custody signal	[RFC5050]
6,7	3	Unassigned	
6	4	Aggregate Custody Signal	[CCSDS-BP]
6,7	5-15	Unassigned	

Bundle Protocol Version	Value	Description	Reference
7	16-65535	Unassigned	
7	greater than 65535	Reserved for Private or Experimental Use	This specification

Next Steps

- Requesting the DTN WG to adopt this document.
- This would eventually be in a cluster with the ACME document registering the new code point.
- The BIBE document would also eventually need code points.

JOHNS HOPKINS
APPLIED PHYSICS LABORATORY

BPSec COSE Context

IETF 113 DTN WG

Brian Sipos
JHU/APL

Background

- BPSec and its Default Security Context are usable but intentionally limited in scope:
  - A limited number of symmetric-keyed encryption and MAC algorithms.
  - Defines a variable additional authenticated data (AAD) binding to the block/bundle.
  - No explicit key identifiers are available.
- For internet-facing nodes, possibly as subnetwork gateways, there is a need for PKI-integrated security.
  - This was indicated by IETF SECDIR review of BPSec draft and also discussed as a near-future need by NASA DTN planning group.
- Don’t want to reinvent the wheel, and CBOR Object Signing and Encryption (COSE) already provides syntax and semantics for current and future PKI security.
  - Even COSE (with a restricted profile as used here) still provides a lot of variability, in the same sense that TLS or S/MIME does, which must be managed out-of-band (e.g. don’t use ECC algorithms if security acceptors can’t support it).

Goals for the BPSec COSE Context

- Do not alter BPSec structures or requirements.
  - This is purely an extension within the existing security context mechanism.
- Handle current symmetric-keyed and PKIX algorithms.
  - Leverage existing algorithm definitions.
- Follow algorithm-use and key-use best practices.
  - Avoid key overuse, use random content encryption keys.
  - Allow Diffie-Hellman static-ephemeral algorithms to be used (both Elliptic and Edwards curves).
- Add as little encoded overhead as possible.
- Inherit future gains made by COSE off-the-shelf algorithms.
  - Planning is already underway for hybrid public key encryption (HPKE) and post-quantum cryptography (PQC).

Proposed COSE Context Contents

- One BPSec context codepoint defined to use in BIB and BCB.
- Parameter and result types defined for each BPSec block type:
  - AAD scope parameter (same semantics as Default SC)
  - De-duplicated last-layer COSE header parameters.
  - Integrity results (COSE MAC and Signature)
  - Confidentiality results (COSE Encrypt using AEAD)
- Public keys in context parameters to de-duplicate data.
  - Potential future extensions could provide additional supporting data (e.g. OCSP stapling).
- Full COSE messages contained in each target's result.
  - Reuse COSE message tags as result type codes.
  - Allows an application to use any current or future COSE algorithm types (and combinations).
  - Allows multiple recipients for a single security block (both BIB and BCB).
  - Interoperability requirements are defined in a COSE Profile (next slide).

Interoperability Profile

- Required algorithms for AES-GCM-256, AES key-wrap, and HMAC-SHA2-256.
- Recommended algorithms for Elliptic Curve, Edwards Curve, and RSA signing and key-wrap/key-generation.
- Additional public key material can be included in an “additional header map”, applying to all results in the block.

BPSec Block	COSE Layer	Name	Code	Implementation Requirements
Integrity	1	HMAC 256/256	5	Required
Integrity	1	ES256	-7	Recommended
Integrity	1	EdDSA	-8	Recommended
Integrity	1	PS256	-37	Recommended
Confidentiality	1	A256GCM	3	Required
Confidentiality	2	A256KW	-5	Required
Confidentiality	2	ECDH-ES + A256KW	-31	Recommended
Confidentiality	2	ECDH-SS + A256KW	-34	Recommended
Confidentiality	2	RSAES-OAEP w/ SHA-256	-41	Recommended

Table 5: Interoperability Algorithms

Next Steps

- This is not intended to replace or supersede existing BPSec interoperability contexts in RFC 9173.
- The point here is to allow BPSec in a PKIX environment in the very near term.
  - COSE is a known quantity with existing coding and processing tools.
  - Identifying bundle security purpose and validation of a Node ID within a PKIX certificate are already defined in RFC 9174.
  - An extension to ACME to automate validation of a Node ID is under review.
- Known changes needed:
  - #10 Align AAD encoding with RFC 9173 for consistency.
- Some secondary questions remain, for example:
  - How does a security acceptor handle a BIB signed by a key with a certificate for a different Node ID than the security source? Base BPSec doesn't really deal with identity/authentication logic.
  - Is there a more strict minimum COSE header content? S/MIME makes requirements about full certificate presence, while the current draft allows an "x5t" thumbprint as a placeholder for compact encoding.

JOHNS HOPKINS
APPLIED PHYSICS LABORATORY

Bundle Version Identification

IETF 113 DTN WG

Brian Sipos
JHU/APL

Need For this Document

- RFC 5050 and RFC 9171 define BPv6 and v7 respectively.
- Each encoding places the self-identifying version number in a different location in the message.
  - BPv6 uses a fixed offset and size.
  - BPv7 uses CBOR structure which doesn't place at a fixed offset.
  - Some current tools ***assume*** that BPv7 version *is* at a fixed offset and size, based on example bundles and spot checks with some bundle producers.
- In some systems and networks there can be a need to identify a byte string as a specific bundle version.
  - This may be for inspection, and not (yet) fully decoding the bundle.
- A trivial mechanism to detect v6 vs. other (assumed v7) was drafted in https://datatracker.ietf.org/doc/html/draft-sipos-dtn-udpcl-01
  - This doesn't actually identify the CBOR-encoded version.
  - This doesn't provide any optimization hints for CBOR-encoded bundles.

Proposed Mechanism

- Thankfully, there is no collision between pre-CBOR and CBOR encodings.
  - Also, the pre-CBOR encoding is fully compatible with CBOR uint type.
- Several algorithms possible, depending on agent need.
  - The main requirement is that an agent SHALL NOT assume things about the encoding that are not guaranteed by RFC 9171.
- General purpose, handle any encoding:
  1. Decode the start of the byte string as CBOR.
  2. If it's uint type, that value is the version number.
  3. If it's array type, then decode the first array item. If that's array type, then decode the first array item. If that's uint type, that value is the version number.
- Optimized 1, can fail for non-deterministic or non-preferred encoding:
  - Perform bit mask and comparison for pattern matching. This is effectively a selective CBOR decoder (for certain encodings).
  1. Take first octet, bitwise-AND with 0xE0, and compare to 0x00 to detect v6 and earlier. The octet bitwise-AND with 0x1F is the version number.
  2. Take first octet, bitwise-AND with 0xE0, and compare to 0x80 to detect array. Then take next octet, bitwise-AND with 0xE0, and compare to 0x80 to detect array. Then take next octet, bitwise-AND with 0xE0, compare to 0x00 to detect uint. The octet bitwise-AND with 0x1F is the version if it is less than 24.
- Optimized 2, works only for controlled networks where combinations are limited:
  - Compare the input against fixed byte strings as pattern matching.
  1. 0x06 is a v6 bundle
  2. 0x9F8B07 is a v7 bundle, indefinite framing, non-fragmented, with CRC

  … and so on with the other possible variations.

Next Steps

- Creating a real draft Informational document for this purpose.
- Requesting WG review of the logic and content of the draft.

JOHNS HOPKINS
APPLIED PHYSICS LABORATORY

UDPCL Standardization

IETF 113 DTN WG

Brian Sipos
JHU/APL

Need For this Document

- RFC 7122 gives a brief explanation of BP-over-UDP.
- This explanation has unspecified or weak areas:
  - Makes no mention of IP addressing or relationships of unicast or multicast IP.
  - Allocation of a UDP port is made but no explanation of how the single port number must be used, or how it should relate to things like firewalls or NATs.
  - No explanation of constraints related to UDP itself. New protocols have BCP 145 “UDP Usage Guidelines” to conform to.
  - Assumes that bundle fragmentation is able to handle all situations of content larger than the path MTU.
  - Assumes that the path MTU is *a priori* known or configured.
- There are BP users already needing a fully defined UDPCL.
  - The Draft LunaNet Interoperability Specification mentions the UDPCL in a way that assumes it exists, or will exist, as a standard.

Goals for UDPCLbis

- Maintain backward compatibility: one bundle—one datagram
  - Allow existing implementations to be adapted.
  - Other proposed document for *Bundle Version Identification* would allow BPv6 be kept for full backward compatibility.
- Improve unspecified or weak areas:
  - Defines how unicast and multicast IP are to be used.
  - Make specific UDP port requirements and overall usage requirements.
  - Handle bundles which are barely larger than the path MTU.
- Add long-term extensibility and interoperable CL security:
  - Provide a place to put extensions, keeping the overall send-and-forget strategy
- Further areas for improvement:
  - Bundle packing within a single datagram? This could improve efficiencies for certain workflows.
  - Can a UDP conversation be bidirectional? A “polling” conversation would allow NAT traversal.

Next Steps

- Requesting WG review of the clarifications, especially relative to current implementations, in https://www.ietf.org/archive/id/draft-sipos-dtn-udpcl-01.html
- Bring the draft into alignment with RFC 9174 document structure and editor changes.
- Handle issues documented at https://github.com/BrianSipos/dtn-bpbis-udpcl/issues