Delay-Tolerant Networking
Internet-Draft
Updates: 9174, dtn-bpsec-cose (if approved)
Intended status: Informational
Expires: 17 September 2026

B. Sipos
JHU/APL

16 March 2026

Bundle Protocol Endpoint ID Patterns
draft-ietf-dtn-eid-pattern-07

Abstract

This document extends the Bundle Protocol Endpoint ID (EID) concept into an EID Pattern, which is used to categorize any EID as matching a specific pattern or not. EID Patterns are suitable for expressing configuration, for being used on-the-wire by protocols, and for being easily understandable by a layperson. EID Patterns include scheme-specific optimizations for expressing set membership and each scheme pattern includes text and binary encoding forms; the pattern for the "ipn" EID scheme being designed to be highly compressible in its binary form. This document also defines a Public Key Infrastructure Using X.509 (PKIX) Other Name form to contain an EID Pattern and a handling rule to use a pattern to match an EID.

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 17 September 2026.

Table of Contents

1. Introduction

The Bundle Protocol (BP) Version 7 specification [RFC9171] defines Uniform Resource Identifier (URI) text and Concise Binary Object Representation (CBOR) binary encoding forms of an Endpoint ID (EID). An EID value is used as both a source and a destination for individual Bundles among other uses in BP. In addition to the base protocol, the BP Security (BPSec) specification [RFC9172] uses an EID for security source identity and the TCP Convergence Layer (TCPCL) [RFC9174] uses an EID value for peer identity. BP Agent implementations have necessarily used methods of defining patterns for matching multiple EIDs in order to configure routing, forwarding, and delivery of Bundles, security policy, and convergence layer policy, but these have not yet been standardized and do not have a concise form suitable for on-the-wire messaging.

In much the same way that the Classless Inter-domain Routing (CIDR) mechanism [RFC4632] can be used to aggregate a contiguous and bit-aligned block of IP addresses in a concise unit (encoded as text or otherwise), this concept of EID Pattern is used to aggregate a set of EIDs into a single concise unit. This is valuable because an EID includes both an identifier of the node sending or receiving the Bundle as well as an identifier for the specific service which generated or will process the Bundle. Any EID Pattern can be used both to aggregate EIDs based on node identifier, service identifier, or both.

A purely text-based pattern mechanism such as [W3C-PAT] could handle the general case of matching the text form of EIDs (as URIs) but would not be able to achieve the same level of encoding compression and would not be able to express of exact numeric ranges like the scheme-specific mechanism defined in this document.

The certificate profile and NODE-ID definition of TCPCL [RFC9174] uses the text form of EID to authenticate nodes based on EID. This document defines a Public Key Infrastructure Using X.509 (PKIX) Other Name Form to contain an EID Pattern and a handling rule to use a pattern to match an EID. This allows authenticating an individual EID based on an EID Pattern in much the same way as using a "wildcard" certificate to match a DNS name (see Section 6.3 of [RFC9525]).

1.1. Goals

The text form of an EID Pattern defined in Section 2 is _not_ a URI and is not bound by the character set restrictions imposed for an encoded URI [RFC3986]. This is much the same as a URI template [RFC6570] is also not itself a URI. Although some forms of EID Pattern can contain reserved URI characters, it is not guaranteed that any particular EID Pattern will be intrinsically differentiable from an EID. See Section 5 for details on handling concerns.

For the pattern forms defined in Section 2, the exact-match pattern's text form is identical with its matching EID (with explicitly stated limitations). This behavior is not required or strictly necessary but is a convenient side effect of the text definitions and makes the EID Pattern a proper superset of EID. Because of its structure, used to simplify processing, the CBOR form for EID Pattern will never be identical to or a superset of EID.

One other aspect of this patterning mechanism is that the text form of each scheme-specific pattern is intended to be, in a subjective sense, natural and understandable for the case of a human manually typing patterns into a text document or quick email message; the interpretation of the text pattern needs to "make sense" with minimal training.

In summary, current and new scheme-specific EID Pattern definitions SHALL specify all of the following:

* A logical information model for the scheme-specific pattern.

* Any exceptions or qualifications to the goal of text-form EID being an identity EID Pattern (_i.e._, a text EID will act as a pattern unmodified, and that pattern will match only the original EID).

* Logic for matching a specific EID against the information model.

* Logic for performing set operations with the information model (_i.e._, pattern unions, intersections, and subset comparisons).

* Both text-form and CBOR-form encodings for those scheme-specific information models.

1.2. Scope

This document defines a logical model of pattern matching BP Endpoint IDs and both text and CBOR encoding forms, as well as PKIX extensions to make use of EID Patterns in a public key certificate (PKC).

This document does not define a method of disambiguating an EID from an EID Pattern (in either encoded form) without any other context. Given a pure text or CBOR encoding of an arbitrary value, there needs to be some external context to determine how to interpret it.

This document defines scheme-specific pattern for the "ipn" URI scheme, as its semantics are well-established, while the other currently registered "dtn" scheme lacks well-defined semantics for the structure of its authority part (which would be necessary for wildcard logic).

Although the same EID definitions apply to BP Version 6 [RFC5050] this document does not provide any mechanisms of integrating with that protocol. It is an implementation matter for a BP Agent to use EID Patterns with BP Version 6 and its compressed bundle header encoding (CBHE).

1.3. Use of ABNF

This document defines text structure using the Augmented Backus-Naur Form (ABNF) syntax [RFC5234]. The entire ABNF structure can be extracted from the XML version of this document using the XPath expression:

’//sourcecode[@type="abnf"]’

The following initial fragment defines the top-level rules of this document’s ABNF.

```
; Shared wildcard rules
wildcard = "*"
multi-wildcard = "**"

non-zero-decimal = (%x31-39 *DIGIT)
```

The definition for the rule UTF8-octets is taken from UTF8 [RFC3629]. The definitions for the rule scheme is taken from URI [RFC3986]. The definition for the rule digit is taken from ABNF [RFC5234]. The definition for the rules ipn-decimal and nbr-delim are copied from BP [RFC9171].

1.4. Use of CDDL

This document defines CBOR structure using the Concise Data Definition Language (CDDL) syntax [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 rules of this document's CDDL, which includes rules needed for validating the example CBOR content.

```
start = eid-pattern / embed-eid-pattern
```

The definition for the rule eid-structure is copied from BP [RFC9171].

1.5. Terminology

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

The terms "Endpoint" and "Endpoint ID" in this document refer to the meanings defined in Section 3.1 of [RFC9171].

Other terms specific to this document are the following.

Normalize: This means to change a pattern value while retaining its meaning, for example enforcing logic like de-duplication, case-insensitivity, or domain constraints. The internal representation of a pattern is expected to be normalized, even if an encoded value is not (_e.g._, human-input text form).

Canonicalize: This means to choose a specific encoded form of a pattern value without changing the represented logical value, for example using a preferred ordering or a compressed code point. The text form of encoding has more canonicalization options than the binary form, and human-input text form is expected to not always use canonical choices.

Elide: This is an encoding choice to not represent both the text and integer form of a scheme in the Any-SSP Pattern Item when it is known by the encoder that all possible decoders of the pattern understand that scheme and are expected to re-constitute the elided form.

2. Patterns for BP Endpoint IDs

This document does not define a universal form of EID Pattern, though text forms of EID Patterns do share concepts and rules for wildcard matching (_e.g._, [RFC4592]). Instead, in order to achieve efficiencies in non-text encoding, each EID scheme uses a different form of complex pattern matching. There are also scheme-independent match-all forms that function without a processor needing scheme-specific logic for all possible schemes.

An EID Pattern processor MAY normalize the internal representation of a pattern to an equivalent one without keeping track of the original pattern information or encoding. If an pattern-using application needs to ensure that original encodings are kept, that needs to happen outside of the pattern processor. See Section 4 for recommendations about this need.

2.1. Pattern Set and Pattern Items

The overall concept of this patterning structure is that one "EID Pattern" can be used to match any combination of EIDs. This is accomplished by a single pattern being composed of independent pattern items, each with specific rules and syntax.

The conceptual model of the EID Pattern is as a sequence of pattern items, some of which are general purpose and some with scheme-specific content. This sequence is ordered in order to make translating between forms deterministic, as each encoding form necessarily has a specific order of items. To ensure interoperability, all implementations SHALL support at least 10 items per pattern. Implementations SHOULD support at least 100 items per pattern.

Although the encoding forms are necessarily ordered, the matching logic for an EID Pattern is independent of the order of its items. An EID pattern SHALL be considered to match an EID if any of its constituent items match the EID. The sequence is also allowed to be empty, which by that definition will match no EIDs.

Because matching against an "any-scheme" item (see Section 2.2) will necessarily make any scheme-specific patterns redundant, the text and CBOR forms of the EID pattern have a compressed form of any-scheme matching and disallow combining the any-scheme pattern with other items.

2.1.1. Text Form

The text form of the EID pattern is the following ABNF rule, which uses the URI reserved character "|" to delimit items in the sequence. This rule also allows a pattern to be empty, having no items.

```
eid-pattern = any-scheme-item / eid-pattern-set
; The set is allowed to be empty
eid-pattern-set = [ eid-pattern-item *( "|" eid-pattern-item ) ]
eid-pattern-item = any-ssp-item
; Extension point at eid-pattern-item for scheme-specific rules
; constrained by the structure below, not enforced by ABNF

eid-item-structure = scheme ":" *ssp-char
ssp-char = <UTF8-octets excluding "|">
```

Because the delimiter is used between items, an EID pattern with one item has an identical text form to that item. The correspondence in text form between a single EID and an EID pattern item which matches that single EID SHALL be enforced by any future scheme-specific pattern syntax registered with IANA.

Any current or future text form scheme-specific pattern structure SHALL conform to the existing URI structure of a scheme name (matching the scheme rule from URI) followed by a colon (":") followed by a scheme-specific part. The text form pattern item scheme-specific part SHALL contain UTF8 characters [RFC3629] excluding the separator "|". Because of that constraint, the number of items in a text form pattern can be determined by counting the number of "|" separators present (with a special case for the empty pattern).

Beyond that simple constraint, all other characters (including those prohibited from URI syntax) can be part of a scheme-specific pattern definition. It is RECOMMENDED that future scheme-specific pattern definitions consider human-readability of the scheme-specific character set.

2.1.2. CBOR Form

The CBOR form of the EID pattern is the following, which uses an enveloping array to contain the items. Although the any-scheme pattern includes a compressed encoding, avoiding the outer array, it still follows the conceptual model of a set of items (in which there is allowed only one item). The empty array allows a pattern to be empty, having no items.

```
eid-pattern = any-scheme-item / eid-pattern-set
eid-pattern-set = [* eid-pattern-item]
eid-pattern-item = scheme-pat-item / any-ssp-item

scheme-pat-item = $eid-pat-item .within eid-structure
; Each pattern still follows eid-structure from [RFC9171]
eid-structure = [uri-code: uint, SSP: any]
```

Because there is otherwise always an outer array when a scheme-specific item is present, there is no concept of a "bare" scheme-specific item in the CBOR form and no exact correspondence in binary form between a single EID and an EID pattern item which matches that single EID.

Any current or future CBOR form scheme-specific pattern structure SHALL conform to the existing EID structure of a two-item array containing a scheme number and a scheme-specific part as a single CBOR item.

2.2. Any-Scheme Pattern Item

The simplest pattern item is one which will match any EID of any URI scheme. Because this necessarily disallows scheme-specific logic, the any-scheme pattern has only its identity with no parameters or conceptual structure.

When the any-scheme item is present in an EID pattern, it SHALL be the only item in the pattern. Any other, scheme-specific items would be redundant and unnecessary when combined with the any-scheme item.

The text form of the any-scheme pattern is the following ABNF which matches only the exact text *:**. As defined in Section 2.1, when this text form is present it cannot be combined with other items.

```
any-scheme-item = wildcard ":" multi-wildcard
```

The CBOR form of the any-scheme pattern is the following CDDL which matches only the exact value true. As defined in Section 2.1, when this CBOR form is present it occurs outside of an enveloping array and thus cannot be combined with other items.

```
any-scheme-item = true
```

2.3. Any-SSP Pattern Item

The next most generic pattern item is one which will match any SSP within a specific URI scheme or set of schemes. This includes schemes known to the EID handler as well as schemes (identified by text or integer value) that need not be understood by the EID handler.

As a normalization, when an any-SSP item is present in an EID pattern it SHALL be the only item for the associated schemes. Any other, scheme-specific items would be redundant and unnecessary when combined with the any-SSP item for that same scheme.

As a normalization, only a single any-SSP item SHALL be used to represent all any-SSP matching in a pattern. This avoids redundancy in encoded forms and aids in human understanding when all any-SSP matching is combined in one item.

As a normalization, any schemes identified in any-SSP item SHALL be expanded to include the alternative (text or integer) form of identifier. This enables the _elide_ logic of the encoded forms and serves to support general purpose matching of either text URI forms of EID (using the text form of scheme identifier) or the CBOR forms of EID (using the integer form of identifier).

When present, the canonical ordering of pattern items SHALL place the any-SSP item at the front of the list.

To ensure interoperability, all implementations SHALL support at least 10 schemes (of either text or integer value) per item. Implementations SHOULD support at least 100 schemes per item.

2.3.1. EID Matching

An any-SSP pattern SHALL be considered to match a specific EID when both have the same normalized scheme. Scheme normalization for text EIDs is to convert to a lower-case alphabetic form in accordance with Section 3.1 of [RFC3986]. For schemes which are unknown to the processing entity, the text form of the any-SSP pattern scheme SHALL be used to match text-form EIDs and the integer form of the pattern scheme SHALL be used to match CBOR-form EIDs.

This means that for entities that cannot process a specific (fictional) private-use scheme with value 65536 and name "example", the following text pattern will guarantee proper handling by any entity:

[65536,example]:**

2.3.2. Pattern Set Logic

The parameter for the any-SSP pattern is solely the set of scheme identifiers, so the set logic of this pattern is identical to set logic on those (normalized) identifiers. This can result in a situation where two pattern processors have different matching logic for the same input pattern if they support different EID schemes and are able to expand elided scheme forms differently.

2.3.3. Text Form

The text form of the any-SSP item is the following ABNF, where the scheme-id rule can either be a proper URI scheme or a positive integer value (valid values are restricted by the BP scheme registry [IANA-BP]).

```
any-ssp-item = scheme-set ":" multi-wildcard
scheme-set = scheme-id / ("[" scheme-id *( "," scheme-id ) "]")
scheme-id = scheme / non-zero-decimal
```

The canonical text encoding of a single scheme in an any-SSP item SHALL omit the square brackets. The canonical text encoding of the scheme set in an any-SSP item SHALL place integers first, in ascending order, followed by text names, ordered by length first then by the bytewise lexicographical order of their encoded text. This sorting is identical to the CBOR form in Section 2.3.4.

An encoder MAY elide the integer form of a scheme in a text encoding if it is known that all decoders of the pattern will re-constitute the integer form when normalizing the pattern. It is an implementation matter to determine when to elide during encoding.

2.3.4. CBOR Form

The CBOR form of the any-SSP item is the following CDDL. The first array item conveys no information but because this does not match the eid-structure rule, it is guaranteed to be disambiguated with any current or future scheme-specific $eid-pat-item socket uses.

```
any-ssp-item = [null, + scheme-id]
scheme-id = uint / tstr
```

The canonical CBOR encoding of the scheme set in an any-SSP item SHALL be sorted by the bytewise lexicographic order of their deterministic encodings. This sorting is the same as the deterministic map key ordering of Section 4.2.1 of [RFC8949].

An encoder MAY elide the text form of a scheme in a CBOR encoding if it is known that all decoders of the pattern will re-constitute the text form when normalizing the pattern. It is an implementation matter to determine when to elide during encoding.

2.4. IPN Scheme Pattern Item

As defined in Section 4.2.5.1.2 of [RFC9171] and updated in Section 3 of [RFC9758], IPN scheme EIDs have a SSP which is logically divided into three non-negative integer numeric elements. Because of this, the pattern for IPN scheme EIDs is based on matching a numeric value or range for each element.

For the remainder of this document, the term "IPN pattern" is used as shorthand to mean the EID pattern item used for the "ipn" scheme.

An IPN pattern SHALL logically contain exactly three elements corresponding to the IPN scheme EID elements of:

1. Allocator Identifier, with domain 0 to 2^32-1 inclusive

2. Node Number, with domain 0 to 2^32-1 inclusive

3. Service Number, with domain 0 to 2^64-1 inclusive

The conceptual model of the IPN pattern is that each of the elements of the SSP can be matched as one of:

Specific value: This will match only a single value (as decoded number).

Range: This will match any value contained in a disjoint set of integer intervals.

Wildcard: This will match any valid value.

Within a single element of the IPN pattern, the range intervals SHALL be disjoint and non-contiguous. Any overlapping or contiguity of intervals within a set can be coalesced into a single covering interval with the same meaning. The text form of a range can, but SHOULD NOT, contain overlapping or contiguous intervals. The CBOR form of a range does not allow overlapping or contiguous intervals because of its compressed structure. Within a single element of the IPN pattern, any range interval that includes values outside the element domain SHALL be reduced to fit the domain. The decoder for any form of an IPN pattern SHALL normalize all interval sets to satisfy information model requirements. The decoder for any form of an IPN pattern SHOULD treat the failure of any element of a pattern as a failure to decode the whole pattern.

To ensure interoperability, all implementations SHALL support at least 10 intervals per range in each IPN element. Implementations SHOULD support at least 100 intervals per range in each IPN element.

2.4.1. EID Matching

An IPN pattern SHALL be considered to match a specific EID when both have the same scheme and each element of the the pattern matches the corresponding logical element of the EID SSP. If any element doesn’t match, the whole pattern does not match. Each pattern element SHALL be considered to match according to the following rules:

Specific value: The pattern element SHALL be compared to the EID element as an exact match of integer value.

Range: The pattern element SHALL be considered to match with any EID element value that is contained in any of the finite intervals of the range.

Wildcard: The pattern element SHALL be considered to match with any EID element value.

Because these are dealing with numeric values in an information model, the matching occurs after any encoding-specific normalization (_i.e._ it’s not a text pattern for the text encoding, the matching is performed within the information model of the SSP).

2.4.2. Pattern Set Logic

One benefit of using an EID pattern with an information model of a sequence of numbers or ranges is that performing set logic such as intersection or containment is straightforward. For set logical behavior, the "specific value" case is treated as a singleton set and the wildcard case is treated as the unbounded-interval.

Two IPN patterns are equivalent if their matching EID sets are identical. Two IPN patterns intersect if all of their corresponding elements intersect, and the intersection of each element range can be readily computed using multi-interval set logic. Likewise, one IPN pattern is a subset (or proper subset) of another pattern if all of the elements is a subset (or proper subset) of the other’s corresponding element.

2.4.3. Text Form

The text form of the IPN pattern conforms to the ABNF in Figure 1, which is a superset of the IPN scheme itself as defined in Section 4 of [RFC9758] but with a different structure. Each element is separated by the same character "." as in the IPN URI scheme. This pattern uses reserved URI characters of "[" and "]" (see Section 2.2 of [RFC3986]) to indicate the presence of a range set for a element, the character "," to separate the intervals of a range, the character "-" to indicate an interval within the set, and the character "+" to indicate a half-finite interval.

The enveloping characters "[" and "]" SHALL indicate the presence of a range of possible values for that element. The logical structure and ABNF below disallows the possibility of nested ranges. Within each range, the character "," SHALL separate multiple numeric intervals within the range. The presence of a completely empty interval (_e.g._, "[]" or "[,3]") is disallowed by the ABNF below and SHALL be treated as invalid. If an interval contains a single numeric value it SHALL be treated is a length-one range. If an interval contains two numeric values separated by a "-" character, the interval SHALL be treated as inclusive of both values. The lower bound of the interval is expected be on the left side of the "-" separator, but decoders SHALL handle both possible orderings of interval bounds. If an interval contains a single numeric value followed by the half-finite "+" character it SHALL be treated as having the lower bound of that value and the upper bound as the largest value in the domain of that element. When encoding an interval, if its last included value is the largest value in the domain of that element the canonical half-finite form SHALL be used.

```
eid-pattern-item =/ ipn-pat-item
ipn-pat-item = "ipn:" (ipn-pat-ssp3 / ipn-pat-ssp2)
; Full pattern for three elements
ipn-pat-ssp3 = ipn-part-pat
               nbr-delim ipn-part-pat
               nbr-delim ipn-part-pat
; Each element in the pattern
ipn-part-pat = ipn-decimal / ipn-range / wildcard

; Same normalized form as IPN scheme itself
ipn-decimal = "0" / non-zero-decimal
nbr-delim = "."

ipn-range = "[" ipn-interval *( "," ipn-interval ) "]"
ipn-interval = ipn-decimal [ ("-" ipn-decimal) / "+" ]

; Accept a single EID in two-element text form
ipn-pat-ssp2 = ("!" / ipn-decimal) nbr-delim ipn-decimal
```

Figure 1: IPN Pattern ABNF Schema

When decoding a two-element IPN pattern, the first element SHALL be treated as a fully-qualified node number (FQNN) in accordance with Section 3.3.1 of [RFC9758] and decomposed into separate allocator and node number elements each matching a specific value. When decoding a two-element IPN pattern, the first element text "!" SHALL be treated as the LocalNode FQNN tuple (0, 2^32 - 1) in accordance with Section 3.4.2 of [RFC9758].

> | The FQNN in two-element form has a domain of 0 to 2^64 - 1
> | inclusive.

When decoding, a pattern processor does not need to keep track of how many elements the original pattern used; the pattern itself always has three elements as defined in Section 2.4.

The canonical text form of an IPN pattern SHALL use three elements. The canonical text form SHALL NOT contain any overlapping or contiguous intervals. The canonical text form SHALL order all intervals in ascending numeric order. The canonical text form SHALL encode all intervals with the lower bound before the upper bound.

2.4.4. CBOR Form

The CBOR form of the IPN pattern conforms to the CDDL in Figure 2. Just as in the IPN URI scheme the pattern scheme identifier is 2, the first elements of the SSP identify the node and the last element identifies the service.

Each of the IPN pattern elements SHALL be CBOR encoded as follows:

Specific value: A number corresponding to the uint rule.

Range: An array item corresponding to the ipn-range rule.

Wildcard: The true item.

The wildcard sentinel values have no intrinsic meaning and were simply chosen to be one-octet-encoded special items. The encoding of ranges is a compressed form in which each logical pair of values in the range array indicates:

1. The least included value if present in the first pair or the width of the excluded interval (after the previous included interval) for later pairs.

2. The width of this included interval, which is omitted if the last interval extends to the largest value for that element (which means it only applies to the last pair).

The "width" in these intervals is defined as the difference between the first and last value in the interval, meaning a zero-width interval contains exactly one value. Another way to interpret the range array is that each number after the first (least) value indicates the width of alternating _included_ and _excluded_ intervals for the range, as depicted in Figure 3 where the width of the last included interval can be omitted (extending the interval to the whole domain).

```
$eid-pat-item /= [
  scheme-num: 2,
  SSP: ipn-ssp
]
ipn-ssp = [
  ; This CDDL representation does not restrict to the
  ; true domain of each element.
  3*3 ipn-part-pat,
]
ipn-part-pat = ipn-val / ipn-range / true

ipn-range = [
  ; least included value
  least: ipn-val,
  * (
    ; width of included interval
    incl: ipn-width
    ; width of excluded interval
    excl: ipn-width,
  ),
  ; Width of last included interval or
  ; an implicit half-finite last included interval
  ? incl: ipn-width
]

ipn-val = uint
ipn-width = uint
```

Figure 2: IPN Pattern CDDL Schema

For the compressed form of indicating a half-finite last included interval, a definite-length array head can be used as an early hint because the number of array items will be even when the last interval has an explicit width and odd when it does not. When encoding a range, if its last included value is the largest value for that element the canonical half-finite form SHALL be used.

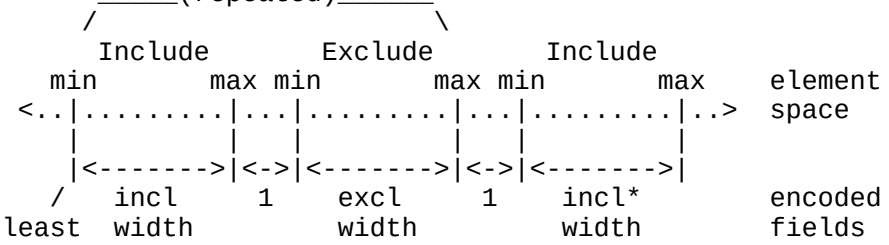

Figure 3: IPN Range Encoding

The most simple finite range is a single value, for example the value 10 can be encoded as [10,0] (which is a special case because this can also use the non-range specific value encoding). The most simple disjoint range containing the values 2 and 4 is encoded as [2,0,0,0] because each included interval is width zero and each excluded between them is also width zero. The largest possible range covering the entire 64-bit number space can be encoded as [0,0xFFFFFFFFFFFFFFFF] or the more concise canonical form of [0] (and even this is a special case because this can also be represented by the wildcard value true, but that is a normalization decision not a coding one).

3. PKIX Certificate Profile Update

This document expands upon the PKIX profile of TCPCLv4 [RFC9174] to allow an EID Pattern in any certificate where an Node ID is required or allowed.

3.1. New Other Name Form

This document defines a PKIX Other Name Form identifier, id-on-bundleEIDPattern in Appendix A; this identifier can be used as the type-id in a Subject Alternative Name (SAN) entry of type otherName. The BundleEIDPattern value associated with the otherName type-id id-on-bundleEIDPattern SHALL be an EID Pattern text form, encoded as an UTF8String, with a scheme that is present in the IANA "Bundle Protocol URI Scheme Types" registry [IANA-BP].

> | The other name form is encoded as an UTF8String because it is
> | _not_ a URI and does not have all of the character restrictions
> | of a URI.

3.2. New Identifier Type

This specification defines an EID-PATTERN-ID of a certificate as being the Subject Alternative Name entry of type otherName with a name form of BundleEIDPattern and a value limited to an EID Pattern text form. An entity SHALL ignore any entry of type otherName with a name form of BundleEIDPattern and a value that is some text other than an EID Pattern.

The EID-PATTERN-ID is similar to the NODE-ID as defined in Section 4.4.1 of [RFC9174] but can match many different and distinct Endpoint IDs. URI matching of an EID-PATTERN-ID SHALL use the scheme-specific EID matching logic defined in this specification. An EID Pattern scheme can refine this matching logic with rules regarding how Endpoint IDs within that scheme are to be compared with the issued EID-PATTERN-ID.

As an augmentation of the TCPCL certificate profile in Section 4.4.2 of [RFC9174]: Unless prohibited by CA policy, a TCPCL end-entity certificate SHALL contain either a NODE-ID or an EID-PATTERN-ID that authenticates the node ID of the peer. All other requirements of that certificate profile are unchanged by this document.

3.3. New Name Constraints Logic

This document defines a logic for using EID Pattern(s) within the Name Constraints extension of Section 4.2.1.10 of [RFC5280] for CA certificates. Because the EID Pattern does not define a general-purpose subset logic, a Name Constraints with an EID Pattern cannot directly constrain subordinate SANs with EID or EID Pattern items so has no effect on path validation (see Section 6 of [RFC5280]). It is instead used to constrain the ultimate identity validation (see Section 6 of [RFC9525] and Section 4.4.4 of [RFC9174]) for Node IDs specifically and any future validation of EIDs more generally as defined below.

As an augmentation of the TCPCL peer authentication in Section 4.4.4.3 of [RFC9174]: When performing a validation of a Node ID against an end entity certificate with NODE-ID or EID-PATTERN-ID, the validation SHALL also validate the Node ID based on all of the CA certificates in the path which contain a Name Constraints extension itself containing an Other Name Form of id-on-bundleEIDPattern. That match SHALL consider both the permitted and excluded subtrees of the Name Constraints in accordance with Section 4.2.1.10 of [RFC5280].

Due to the nature of matching any possible EID, a Name Constraints extension SHOULD NOT contain an BundleEIDPattern with the match-all pattern *:** as this serves no purpose. Including a match-all pattern in the included subtrees does not add any value and including it in the excluded subtrees is functionally the same thing as disallowing the presence of the id-kp-bundleSecurity Extended Key Usage.

When issuing CA or end entity certificates, a CA limited by Name Constraints containing BundleEIDPattern values MAY make use of scheme-specific subset logic to determine that the combination of end entity SAN and CA Name Constraints will not validate any possible Node ID and refuse to issue the requested certificate. For example, a root CA constrained with an included subtree of ipn:0.*.* could disallow issuing a subordinate intermediate CA with a constrained included subtree of ipn:** because it isn’t a proper subset of its parent constraint, or could disallow issuing an end entity certificate with a SAN identity of ipn:977000.2.3 because it is guaranteed to not pass Node ID validation. The refusal or not to issue such subordinate certificates does not affect the ultimate

validation of the Node ID but does make it less likely for certificates to be used by an end entity which will never succeed at Node ID validation.

4. Enveloping Considerations

When an EID pattern is enveloped into a data store or protocol data unit, it is important to avoid requiring the processor of that containing context to understand the nuances of EID Pattern syntax. For the text form of EID Patterns this is straightforward because the encoded text string can simply be handled without concern for its contents. The use of an EID Pattern as a PKIX Other Name Form in Section 3 makes use of this strategy.

For the binary form of EID Patterns, when the encoded item is not handled as a simple byte string it is RECOMMENDED to embed the EID Pattern within a CBOR byte string as a single item. This is formalized by the following CDDL.

```
embed-eid-pattern = bstr .cbor eid-pattern
```

Embedding in a byte string this way allows BP EID Pattern-unaware processors to handle it without concern for its syntax or validity. Using a single-item embedding ensures that the number of pattern items contained is still available to decoders in the eid-pattern array head.

A similar recommendation is provided here for enveloping EIDs themselves, which is not discussed in BP [RFC9171] so this document does not formally update that specification. For the binary form of EIDs, when the encoded item is not handled as a simple byte string it is RECOMMENDED to embed the EID within a CBOR byte string as a single item. This is formalized by the following CDDL.

```
embed-eid-structure = bstr .cbor eid-structure
```

Embedding in a byte string this way allows BP EID-unaware processors to handle it without concern for its syntax or validity. Although this adds some redundancy to the encoding because the eid-structure is always a two-element array, it is limited to the single byte of the its array head. This is also consistent with how the existing Previous Node block-type-specific data content is defined in Section 4.4.1 of [RFC9171].

5. Security Considerations

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

5.1. Denial of Service Possibilities

Because EID patterns are expected to be used as external input to a BP Agent, there is the possibility of an encoded value to either be not-well-formed or well-formed but purposefully resource intensive. Either of these cases can cause resource exhaustion of a BP Agent and thus deny resources to normal data services provided by the Agent. These considerations are not exclusive to the EID Pattern syntax, and apply just as well to any other external input data.

An example of a malicious not-well-formed pattern is a CBOR form that contains an array with a definite length that is larger than the actual encoded pattern. A trivial decoder might attempt to pre-allocate memory based on the array head and exhaust its memory. A mitigation of this is to have bounds on acceptable lengths/sizes within the CBOR decoder.

An example with a well-formed pattern is to simply encode a pattern set with thousands or millions of arbitrary, but valid, items. A decoder could attempt to fully decode this valid pattern but by doing so exhaust its memory. A similar mitigation here is to bound acceptable sizes above the CBOR layer within the pattern decoding.

5.2. Misinterpretion of Encoded Values

It is critical for applications handling EIDs and EID Patterns to positively distinguish between the two based on the context in which the value is being used. For PKIX Subject Alternative Name this is distinguished by the different Other Name forms. An EID which is inappropriately interpreted as an EID Pattern could allow an attacker to elevate access depending upon other aspects of the system being accessed.

CAs which issue certificates containing EID Patterns need to consider the implications of an overly-broad pattern in the same way that current Web PKI CAs manage certificates with wildcard DNS-IDs. This is discussed for DNS-IDs in Section 7.1 of [RFC9525].

Although the reserved characters "[" and "]" are disallowed within the URI authority and path segments by [RFC3986] there are still URI processors which could be lax about enforcing that restriction and could allow an EID pattern to be decoded in a place where an actual EID is expected. This could allow unwanted side-effects when the EID is handled by a BP Agent.

The URI authority part and path segments are percent-encoded text and need to be handled by EID processors as such for both pattern matching and equality comparison. Additionally, for the IPN scheme there are numeric values that need to be handled as such for pattern matching and comparison.

5.3. Expansion of Elided Any-SSP Schemes

The requirements for the any-SSP pattern in Section 2.3 allow an encoder (which in some cases is expected to be a human) to elide forms of specific scheme identifiers based on knowledge (or assumptions) about decoders of the pattern. If the knowledge or assumptions are not valid, there is the possibility that

For example, given the fully-identified pattern

[65536,example]:**

if this pattern passes through a CBOR encoding which elides the text form, the pattern is logically reduced to

[65536]:**

and any pattern processor which does not know the name "example" for this private use scheme will not be able to match text form EID values.

As somewhat of a mitigation of this issue, when used as external input to a BP Agent through CBOR encoded pattern the text-elided form will still always be able to properly match CBOR encoded EID values within Bundles themselves.

6. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of code points in existing registries in accordance with BCP 26 [RFC8126].

6.1. Bundle Protocol URI Scheme Types

This specification re-uses the "Bundle Protocol URI Scheme Types" registry within the "Bundle Protocol" registry group [IANA-BP] for the CBOR encoding of EID Patterns and adds an informative column "EID Pattern Reference" as in the following table.

Specifications of new EID Pattern schemes SHALL define all of the required items from Section 1.1 to ensure that pattern behavior is consistent across different schemes.

Value	Description	...	EID Pattern Reference
2	ipn		Section 2.4 of [This specification]

Table 1: Bundle Protocol URI Scheme Types

6.2. Object Identifier for PKIX Other Name Forms

IANA has created, under the "Structure of Management Information (SMI) Numbers" registry group [IANA-SMI], a registry titled "SMI Security for PKIX Other Name Forms". This other name forms table is updated to include a row for containing an Endpoint ID Pattern as in the following table.

Decimal	Description	References
ON-TBA	id-on-bundleEIDPattern	[This specification]

Table 2: PKIX Other Name Forms

The formal structure of the associated other name form is in Appendix A. The use of this form is defined in Section 3.

7. References

7.1. Normative References

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

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

[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>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[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>.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <https://www.rfc-editor.org/info/rfc4632>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[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>.

[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>.

[RFC9171] Burleigh, S., Fall, K., and E. Birrane, III, "Bundle Protocol Version 7", RFC 9171, DOI 10.17487/RFC9171, January 2022, <https://www.rfc-editor.org/info/rfc9171>.

[RFC9174] Sipos, B., Demmer, M., Ott, J., and S. Perreault, "Delay-Tolerant Networking TCP Convergence-Layer Protocol Version 4", RFC 9174, DOI 10.17487/RFC9174, January 2022, <https://www.rfc-editor.org/info/rfc9174>.

[RFC9525] Saint-Andre, P. and R. Salz, "Service Identity in TLS", RFC 9525, DOI 10.17487/RFC9525, November 2023, <https://www.rfc-editor.org/info/rfc9525>.

[RFC9758] Taylor, R. and E. Birrane III, "Updates to the ’ipn’ URI Scheme", RFC 9758, DOI 10.17487/RFC9758, May 2025, <https://www.rfc-editor.org/info/rfc9758>.

[X.680] ITU-T, "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2015, August 2015, <https://www.itu.int/rec/T-REC-X.680-201508-I/en>.

7.2. Informative References

[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>.

[RFC4592] Lewis, E., "The Role of Wildcards in the Domain Name System", RFC 4592, DOI 10.17487/RFC4592, July 2006, <https://www.rfc-editor.org/info/rfc4592>.

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

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <https://www.rfc-editor.org/info/rfc5912>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <https://www.rfc-editor.org/info/rfc6570>.

[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>.

[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>.

[RFC9172] Birrane, III, E. and K. McKeever, "Bundle Protocol Security (BPSec)", RFC 9172, DOI 10.17487/RFC9172, January 2022, <https://www.rfc-editor.org/info/rfc9172>.

[github-ricktaylor-hardy]
Taylor, R., "BPv7 DTN server implementation",
<https://github.com/ricktaylor/hardy>.

[W3C-PAT] W3C, "URI Pattern Matching for Groups of Resources", June 2006,
<https://www.w3.org/2005/Incubator/wcl/matching.html>.

Appendix A. ASN.1 Module

The following module formally specifies the BundleEIDPattern structure and its Other Name form in the syntax of ASN.1 [X.680]. This specification uses the ASN.1 definitions from PKIX [RFC5912] with the 2002 ASN.1 notation used in that document.

```
<CODE BEGINS>
DTN-EIDPATTERN-2025
  { iso(1) identified-organization(3) dod(6)
    internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-dtn-eidpattern-2025(MOD-TBA) }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

IMPORTS
  OTHER-NAME
  FROM PKIX1Implicit-2009 -- [RFC5912]
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-implicit-02(59) }

  id-pkix
  FROM PKIX1Explicit-2009 -- [RFC5912]
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-explicit-02(51) } ;

id-on OBJECT IDENTIFIER ::= { id-pkix 8 }

DTNOtherNames OTHER-NAME ::= { on-bundleEIDPattern, ... }

-- The otherName definition for Bundle EID Pattern
on-bundleEIDPattern OTHER-NAME ::= {
    BundleEIDPattern IDENTIFIED BY { id-on-bundleEIDPattern }
}

id-on-bundleEIDPattern OBJECT IDENTIFIER ::= { id-on ON-TBA }

-- Encoding allows URI reserved characters
BundleEIDPattern ::= UTF8String

END
<CODE ENDS>
```

Appendix B. Examples

B.1. IPN Patterns

This section contains examples specific to the IPN pattern of Section 2.4.

B.1.1. Exact Match

This trivial example matches only one EID (which itself has the same text and CBOR forms)

ipn:0.3.4

which has a CBOR form of:

[[2, [0, 3, 4]]]

B.1.2. Wildcards

This example matches all service numbers on a single node

ipn:0.3.*

which has a CBOR form of:

[[2, [0, 3, true]]]

This example matches all default-authority nodes with the same service number

ipn:0.*.4

which has a CBOR form of:

[[2, [0, true, 4]]]

B.1.3. Range Match

This example includes a single range over the service numbers ipn:0.3.0 to ipn:0.3.19 inclusive as

ipn:0.3.[0-19]

which has a CBOR form of:

[[2, [0, 3, [0, 19]]]]

This example includes an offset range over the service numbers ipn:0.3.10 to ipn:0.3.19 inclusive as

ipn:0.3.[10-19]

which has a CBOR form of:

[[2, [0, 3, [10, 9]]]]

This example includes multiple ranges of service numbers ipn:0.3.0 to ipn:0.3.4 and ipn:0.3.10 to ipn:0.3.19 inclusive as

ipn:0.3.[0-4,10-19]

which has a CBOR form of:

[[2, [0, 3, [0, 4, 4, 9]]]]

B.1.4. Normalization

These examples show normalization of specific patterns.

An overlapping or contiguous pattern such as one of the following

ipn:0.3.[0-9,10-19]
ipn:0.3.[0-15,10-19]
ipn:0.3.[10-19,0-9]

can be normalized to the equivalent pattern

ipn:0.3.[0-19]

An unordered pattern such as

ipn:0.3.[10-19,0-4]

can be normalized to the equivalent pattern

ipn:0.3.[0-4,10-19]

A pattern where a range covers more than the domain of a element, as in

ipn:977000.[10000-5000000000].*

can be normalized to the equivalent (and smaller) pattern

ipn:977000.[10000-4294967295].*

A pattern where a range covers the same values as a wildcard would, as in

ipn:977000.[0-4294967295].*

can be identified and normalized to the equivalent pattern

```
ipn:977000.*.*
```

B.1.5. Two-Element Text Form Normalization

The two-element text form can contain only a single EID and cannot have a range or wildcard element.

The two-element LocalNode EID can appear as either of the following

```
ipn:4294967295.0
ipn:!.0
```

these are normalized into the equivalent three-element form of

```
ipn:0.4294967295.0
```

This example includes a range over the FQNN between the limits of 4196183048192100 to 4196183048192500 inclusive, which is decomposed into the equivalent three-element pattern

```
ipn:977000.[100-500].*
```

which has a CBOR form of:

```
[[2, [977000, [100, 400], true]]]
```

The next example has a range over the FQNN which spans multiple allocator IDs between the limits of 4196183048192100 to 4196191638126692 inclusive, which is decomposed into one possible equivalent pattern

```
ipn:977000.[100+].*|ipn:977001.*.*|ipn:977002.[0-100].*
```

which has a CBOR form of:

```
[
  [2, [977000, [100], true]],
  [2, [977001, true, true]],
  [2, [977002, [0, 100], true]]
]
```

B.1.6. Canonicalization

These examples show canonicalization of specific patterns.

When an interval has descending bounds such as

```
ipn:0.3.[10-0]
```

can be canonicalized to the equivalent pattern

ipn:0.3.[0-10]

When the end of an interval is the largest value of the corresponding element, as in

ipn:977000.[10000-4294967295].*

the last value of the last interval can be canonicalized to the pattern

ipn:977000.[10000+].*

which does not affect the information model but makes the encoded form shorter (and more understandable to a human).

B.1.7. Disjoint Ranges

A single IPN-scheme item is not always able to encode a full desired pattern. In these cases multiple items will need to be combined, such as in the following text pattern

ipn:0.*.*|ipn:977000.*.0

which has a CBOR form of:

```
[
  [2, 0, true, true],
  [2, [977000, true, 0]]
]
```

B.2. Combined Patterns

This section contains examples of patterns combining items.

B.2.1. Match None

This trivial example does not match any EID, and this is the preferred way to "match none." It's text form is empty (so not represented here) and its CBOR form is the empty array

```
[]
```

B.2.2. Match All

This trivial example matches any possible EID, and this is the preferred way to "match all." It's text form is:

*:**

and its CBOR form is:

true

B.2.3. Any-SSP Match

These two examples match any ipn-scheme EID, either fully-identified as:

[2,ipn]:**

or integer-elided as:

ipn:**

and both have the following CBOR form either fully-identified as:

[[null, 2, "ipn"]]

or text-elided as:

[[null, 2]]

B.2.4. Multiple Item Match

This example combines items with different schemes together in one pattern, it will match dtn:** and ipn:0.3.4 It’s text form is:

dtn:**|ipn:0.3.4

and its CBOR form is:

```
[
  [null, 1],
  [2, [0, 3, 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-ricktaylor-hardy].]

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.

A trial implementation in Rust language of the EID Pattern encoding and decoding and EID matching logic is present as part of the full BP Agent of Hardy [github-ricktaylor-hardy]. This repository includes unit test vectors for verifying pattern handling.

Acknowledgments

Pattern expressiveness is based on use case examples provided by Carlo Caini and Lucien Loiseau. Prototyping of and validation for the utility of these patterns was performed by Rick Taylor.

Author’s Address

Brian Sipos
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
United States of America
Email: brian.sipos+ietf@gmail.com