Workgroup: Delay-Tolerant Networking
Internet-Draft: draft-sipos-dtn-eid-pattern-02
Published: 8 April 2024
Intended Status: Informational
Expires: 10 October 2024
Authors: B. Sipos
JHU/APL

# Bundle Protocol Endpoint ID Patterns

## 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 CBOR encoding forms; the pattern for the "ipn" EID scheme being designed to be highly compressible in its CBOR 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 10 October 2024.

## Copyright Notice

**Table of Contents**

## 1. Introduction

The Bundle Protocol (BP) Version 7 specification of [RFC9171] defines Uniform Resource Identifier (URI) text and Concise Binary Object Representation (CBOR) encoding forms of an Endpoint ID (EID) which is used as both a source and a destination for individual bundles, the BP Security specification of [RFC9172] uses EIDs as security sources, and the TCP Convergence Layer (TCPCL) of [RFC9174] uses EIDs for peer identification. 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 of [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 especially 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 [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 Section 6.4.3 of [RFC6125] to match a DNS name.

### 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 in [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 4 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 should "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.

* 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.

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 must be some external context to determine how to interpret it.

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 bundles and their compressed bundle header encoding (CBHE).

### 1.3. Use of ABNF

This document defines text structure using the Augmented Backus-Naur Form (ABNF) of [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-number = (%x31-39 *DIGIT)
```

From the document [RFC3986] the definition is taken for pchar and scheme. From the document [RFC5234] the definition is taken for digit. From the document [RFC9171] the definition is taken for nbr-delim.

### 1.4. 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, which includes the example CBOR content.

```
start = eid-pattern
```

From the document [RFC9171] the definition is taken for eid-structure.

### 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.

## 2. Patterns for 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.

### 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 scheme-specific rules and syntax.

The conceptual model of the EID Pattern is as a non-empty sequence of scheme-specific pattern items. This sequence is ordered in order to make translating between forms deterministic, as each encoding form necessarily has a specific order of items.

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.

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 others.

The text form of the EID pattern is the following, which uses the URI reserved character "|" to delimit items in the sequence. Because the delimiter is used between items, an EID pattern with one item has an identical text form to that item.

```
eid-pattern = any-scheme-item / eid-pattern-set
eid-pattern-set = eid-pattern-item *( "|" eid-pattern-item )
eid-pattern-item = scheme-pat-item / any-ssp-item
; Limited to these schemes for now
scheme-pat-item = ipn-pat-item / dtn-pat-item
```

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). Because there is otherwise always an outer array, there is no concept of a "bare" scheme-specific pattern item in the CBOR form.

```
eid-pattern = any-scheme-item / eid-pattern-set
eid-pattern-set = [1* eid-pattern-item]
eid-pattern-item = scheme-pat-item / any-ssp-item
; Each pattern still follows eid-structure
scheme-pat-item = $eid-pat-item .within eid-structure
```

### 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 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 exact value. 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. This includes schemes known to the EID handler as well as schemes by enumerated integer that need not be understood by the EID handler.

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

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

```
any-ssp-item = (scheme / non-zero-number) ":" multi-wildcard
```

The CBOR form of the any-SSP pattern is the following CDDL. 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 = (uint .gt 0) / tstr
```

#### 2.3.1. EID Matching

An any-SSP pattern **SHALL** be considered to match a specific EID when both have the same normalized scheme. For schemes which are known to the processing entity, the integer form **SHALL** be the normalized form. For schemes which are unkown 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 pattern will guarantee proper handling by any entity:

```
example:**|65536:**
```

### 2.4. DTN Scheme Pattern Item

As defined in Section 4.2.5.1.1 of [RFC9171], DTN scheme EIDs have an authority (node name) part and a sequence of path (service demux) segment components. Combining these components together, the whole EID SSP is treated as a sequence of these unstructured text components. Because of the lack of more specific structure, outside of match-all wildcards only a generic pattern matching mechanism like a regular expression can be used.

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

The conceptual model of the DTN pattern is that the node name and the sequence of path segments can be matched as one of:

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

**Regular expression:** This will match a decoded text value based on a (possibly anchored) regular expression.

**Single-segment wildcard:** This will match an individual path segment.

**Multi-segment wildcard:** For the node name this will match any valid value. For the path segment this will match any number of segments of any value.

A DTN pattern **SHALL** contain at least two components: the first for the node name and the others for the service demux. A DTN pattern **SHALL** contain no more than one multi-segment wildcard component. If present, a DTN pattern **SHALL** only contain a multi-segment wildcard in its last (demux path segment) component.

The reason for using a multi-segment wildcard in the node name part is to allow for a future enhancement of this pattern method to handle components within the node name (similar to the sequence of labels within a DNS name). For now the multi-segment wildcard within a node name behaves equivalently to a single-segment wildcard because the node name is not decomposed into internal components.

### 2.4.1. EID Matching

When matching a DTN pattern any query or fragment parts of an EID **SHALL** be ignored and not treated as comparison components. A DTN pattern **SHALL** be considered to match a specific EID when both have the same scheme, the pattern has the same number of components as the EID, and each component of the the pattern matches the corresponding component of the EID SSP. If the number of components differ or if any component doesn't match, the whole pattern does not match. Each pattern component **SHALL** be considered to match according to the following rules:

**Specific value:** The pattern component **SHALL** be compared with the EID component after both are percent-decoded in accordance with Section 2.1 of [RFC3986] and UTF-8 decoded in accordance with [RFC3629].

**Regular expression:** The pattern component **SHALL** be percent-decoded and UTF-8 decoded then interpreted as a regular expressing in accordance with [ECMA262]. The EID component **SHALL** be percent-decoded and UTF-8 decoded. The regular expression **SHALL** then be compared with the decoded EID component.

**Single-segment wildcard:** The pattern component **SHALL** be considered to match with any EID component, if present, including an empty component.

**Multi-segment wildcard:** The pattern component **SHALL** be considered to match with any number of EID components, including zero EID components.

Because these are dealing with text values in an information model, the matching occurs in the percent-encoding normalized or percent-decoded domain (*i.e.* it's not a pattern for the encoded URI, the matching is performed within the information model of the SSP).

#### 2.4.2. Pattern Set Logic

Because of the arbitrarily complex nesting rules allowed by regular expressions, and the multiple techniques available for different expressions to match the same subsets of text, DTN pattern sets can only be consistently computed when the node-name or demux path segments are either exact-text matches or one of the match-all wildcards.

Users of the DTN pattern **SHALL** have a mechanism to perform set logic with specific value and wildcard components. EID Pattern processors **MAY**, but cannot be assumed to, have a mechanism to perform set logic on regular expression components.

#### 2.4.3. Text Form

The text form of the DTN pattern conforms to the ABNF in Figure 1. The authority begins with the same string "//" and authority and demux components are separated by the same character "/" as in the DTN URI scheme.

This pattern uses reserved URI characters of "[" and "]" (see Section 2.2 of [RFC3986]) to indicate the presence of a regular expression for a component. This allows completely disambiguating a DTN pattern from a specific DTN EID when a regular expression or wildcard is present. Because neither of those are required to be present in a DTN pattern and the asterisk "*" is a valid path segment character, the considerations of Section 4 still always apply to decoding text as EID Pattern versus an EID.

```
dtn-pat-item = "dtn:" dtn-ssp
dtn-ssp = dtn-wkssp-exact / dtn-fullssp

; A node-name authority with some number of demux path segments
dtn-fullssp = "//" dtn-authority-pat "/" dtn-path-pat
dtn-authority-pat = exact / regexp / multi-wildcard
; Only the last path segment is allowed a multi-wildcard
dtn-path-pat = *( dtn-single-pat "/" ) dtn-last-pat
dtn-single-pat = exact / regexp / wildcard
dtn-last-pat = dtn-single-pat / multi-wildcard

; Exact-match text, which excludes gen-delims characters
exact = *pchar
; Regular expression for the whole SSP within the gen-delims brackets
; with an allowance for more regexp characters
regexp = "[" *( pchar / "^" ) "]"

; Exact match for well-known SSP
dtn-wkssp-exact = "none"
```

Figure 1: DTN Pattern ABNF Schema

A concrete use of this text form is illustrated in this example:

```
dtn://node-name/[%5Eanchored]/other%20part/**
      <-- P -->  <--- P --->  <--- P ---->
```

Where the "P" sections are percent-encoded (with no reserved characters) and square brackets unambiguously delimit the expression component. The actual components in this example are the specific value "node", the regular expression "^anchored", and the specific value "other part" and all are UTF-8 and percent-encoded. Further examples are given in Appendix B.1.

Because all of "." "*" "+" and "$" are within the pchar rule, and "^" is added by the regexp rule, it is possible for a less strict encoder (*e.g.* a human writing patterns) to create one similar to dtn://node/[^some.*thing$] and have it still be handled correctly.

#### 2.4.4. CBOR Form

The CBOR form of the DTN pattern conforms to the CDDL in Figure 2. Just as in the DTN URI scheme the pattern scheme identifier is 1, the first component of the SSP identifies the node and the last components identify the service path segments. The well-known SSP **SHALL** be encoded using the same uint value specified for the DTN URI scheme.

Each of the DTN pattern components **SHALL** be CBOR encoded as follows:

**Specific value:**
A text item (not otherwise UTF-8 or percent-encoded) corresponding to the dtn-exact symbol.

**Regular expression:** A tagged regular expression item corresponding to the regexp symbol.

**Single-segment wildcard:** The true item.

**Multi-segment wildcard:** The false item.

The wildcard sentinel values have no intrinsic meaning and were simply chosen to be one-octet-encoded special items. The CBOR form of the DTN pattern is not as compressible as the IPN pattern, but the exact text is not percent encoded and the regular expression tag "regexp" does save one octet per instance.

```
$eid-pat-item /= [
  scheme-num: 1,
  SSP: dtn-ssp
]
dtn-ssp = dtn-wkssp-exact / dtn-fullssp-parts
dtn-fullssp-parts = [
  dtn-authority-pat,
  dtn-path-pat,
]
dtn-authority-pat = dtn-exact / regexp / multi-wildcard
; Only the last path segment is allowed a multi-wildcard
dtn-path-pat = (
  * dtn-single-pat,
  ? multi-wildcard
)
dtn-single-pat = dtn-exact / regexp / wildcard

dtn-exact = tstr
wildcard = true
multi-wildcard = false

; Exact match for well-known SSP
dtn-wkssp-exact = $dtn-wkssp .within uint
$dtn-wkssp /= 0  ; For "none"
```

Figure 2: DTN Pattern CDDL Schema

## 2.5. IPN Scheme Pattern Item

As defined in Section 4.2.5.1.2 of [RFC9171] and updated in [I-D.ietf-dtn-ipn-update], IPN scheme EIDs have a SSP which is logically divided into three integer numeric components. Because of

this, the pattern for IPN scheme EIDs is based on matching a numeric value or range for each component.

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.

The conceptual model of the IPN pattern is that each of the components 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 numeric intervals.

**Wildcard:** This will match any valid value, but not the absence of a value.

An IPN pattern **SHALL** contain exactly three components corresponding to the IPN scheme EID components.

Within a single component 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 intervals because of its compressed form, but does allow contiguous intervals. The decoder for any form of an IPN pattern **SHALL** normalize all intervals sets to satisfy information model requirements. The decoder for any form of an IPN pattern **SHOULD** treat the failure of any component of a pattern as a failure to decode the whole pattern.

A limitation of this mechanism is that there is no intermediate component pattern between a specific set of finite intervals and the match-all (unbounded) wildcard. There is no capability of including an non-finite bounds within any interval. But the components of the IPN scheme itself have finite bounds so a range can be made to capture component values up to and including the EID component bound.

#### 2.5.1. EID Matching

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

**Specific value:**
The pattern component **SHALL** be compared to the EID component as an exact match of decoded numeric value.

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

**Wildcard:** The pattern component **SHALL** be considered to match with any EID component.

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).

### 2.5.2. 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 intersect if all of their corresponding components intersect, and the intersection of each component 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 components is a subset (or proper subset) of the other's corresponding component.

### 2.5.3. Text Form

The text form of the IPN pattern conforms to the ABNF in Figure 3. Each component 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 component, the character "," to separate each range, and the character "-" to indicate the inclusive range within the set. Each of the numeric values within the range is inclusive. If the range does not contain two values it is a length-one range.

The canonical text form of an IPN pattern **SHALL** order all range sets in ascending numeric order.

```
ipn-pat-item = "ipn:" ipn-ssp
; Three logical components
ipn-ssp = ipn-part-pat nbr-delim ipn-part-pat nbr-delim ipn-part-pat
ipn-part-pat = ipn-number / ipn-range / wildcard

; Same normalized form as IPN scheme itself
ipn-number = "0" / non-zero-number
non-zero-number = (%x31-39 *DIGIT)

ipn-range = "[" ipn-interval *( "," ipn-interval ) "]"
ipn-interval = ipn-number [ "-" ipn-number ]
```

Figure 3: IPN Pattern ABNF Schema

#### 2.5.4. CBOR Form

The CBOR form of the IPN pattern conforms to the CDDL in Figure 4. Just as in the IPN URI scheme the pattern scheme identifier is 2, the first components of the SSP identify the node and the last component identifies the service.

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

**Specific value:** A number corresponding to the uint symbol.

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

**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 pair of values in the range indicates:

1. The non-zero offset from the previous one-past-end-of-range, or the offset from zero if there is no preceding range

2. The length of this range, which is inclusive of the first and last contained value so should always be non-zero

Another way to interpret these pairs is that each number indicates the length of alternating "excluded" and "included" intervals for the range.

```
$eid-pat-item /= [
  scheme-num: 2,
  SSP: ipn-ssp
]
ipn-ssp = [
  3*3 ipn-part-pat,
]
ipn-part-pat = uint / ipn-range / true

ipn-range = [ 1* ipn-interval-pair ]
ipn-interval-pair = (
  offset: uint,
  length: uint .gt 0,
)
```

Figure 4: IPN Pattern CDDL Schema

## 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. Any regular expression within the pattern can have direct, non-percent-encoded UTF-8 characters.

### 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 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 matching EIDs and EID Patterns in Subject Alternative Names within subordinate certificates against EID Pattern constraints in the Name Constraints extension of Section 4.2.1.10 of [RFC5280] within CA certificates.

When a Name Constraints extension contains an Other Name Form of id-on-bundleEIDPattern, the associated BundleEIDPattern value **SHALL** be used to match subordinate certificate Subject Alternative Name with Other Name forms of id-on-bundleEID or id-on-bundleEIDPattern. When the subordinate certificate SAN contains an Other Name Form of id-on-bundleEID the matching for a Name Constraints value of an EID Pattern **SHALL** use the scheme-specific EID matching logic defined in this specification. When the subordinate certificate SAN contains an Other Name Form of id-on-bundleEIDPattern the matching for a Name Constraints value of an EID Pattern **SHALL** use the scheme-specific subset logic defined in this specification.

## 4. Security Considerations

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 must manage certificates with wildcard DNS-IDs.

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.

Both URI authority 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 must be handled as such for pattern matching and comparison.

## 5. IANA Considerations

### 5.1. Bundle Protocol URI Scheme Types

This specification re-uses the "Bundle Protocol URI Scheme Types" sub-registry within the "Bundle Protocol" registry [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 |
|---|---|---|---|
| 1 | dtn | | Section 2.4 of [This specification] |
| 2 | ipn | | Section 2.5 of [This specification] |

Table 1

### 5.2. Object Identifier for PKIX Other Name Forms

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

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

Table 2

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

## 6. References

### 6.1. Normative References

**[ECMA262]** European Computer Manufacturers Association, "ECMAScript Language Specification 5.1 Edition", ECMA Standard ECMA-262, June 2011, <http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf>.

**[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>.

**[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>.

**[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>.

**[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>.

**[I-D.ietf-dtn-ipn-update]** Taylor, R. and E. J. Birrane, "Update to the ipn URI scheme", Work in Progress, Internet-Draft, draft-ietf-dtn-ipn-update-10, 21 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-dtn-ipn-update-10>.

**[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>.

### 6.2. Informative References

**[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>.

**[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>.

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

```
<CODE BEGINS>

DTN-EIDPATTERN-2023
  { iso(1) identified-organization(3) dod(6)
    internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-dtn-eidpattern-2023(MOD-TBD) }

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-TBD }

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

END

<CODE ENDS>
```

## Appendix B. Examples

### B.1. DTN Patterns

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

#### B.1.1. Exact Match

This trivial example matches only one EID (which itself has the same text form)

```
dtn://node/service
```

which has a CBOR form of:

```
[[1, ["node", "service"]]]
```

An example of normalized matching is that the pattern

```
dtn://node/service
```

will still match the EIDs dtn://node/ser%76ice and dtn://no%64e/service because each component match is performed in percent-decoded and UTF-8 decoded form.

#### B.1.2. Wildcards

This example matches a single-segment service demux on a single node

```
dtn://node/*
```

which has a CBOR form of:

```
[[1, ["node", true]]]
```

That single wildcard will match the empty demux dtn://node/ but will not match demux paths such as dtn://node/long/name or any more segments.

EDITORIAL NOTE: Do we want the wildcard to actually match the empty segment? Or would it be better to handle that separately so that the above pattern does not match the empty demux?

This example matches all service demux on a single node with a multi-wildcard

```
dtn://node/**
```

which has a CBOR form of:

```
[[1, ["node", false]]]
```

This example matches a service demux with a prefix segment "pre"

```
dtn://node/pre/**
```

which has a CBOR form of:

```
[[1, ["node", "pre", false]]]
```

This example matches all node names having the same service demux

```
dtn://**/some/serv
```

which has a CBOR form of:

```
[[1, [false, "some", "serv"]]]
```

#### B.1.3. Regular Expression Match

This example includes a single regular expression for single-segment service that starts with the letter "a" in the text form of

```
dtn://**/[^a]
```

which has a CBOR form of:

```
[[1, [false, 35("^a")]]]
```

### B.2. IPN Patterns

This section contains examples specific to the IPN pattern of Section 2.5.

#### B.2.1. 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.2.2. 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.2.3. 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, 20]]]]
```

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, 10]]]]
```

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, 5, 5, 10]]]]
```

An overlapping or contiguous pattern such as ipn:0.3.[0-9,10-19] or ipn:0.3.[0-15,10-19] or ipn:0.3.[10-19,0-9] would be normalized to

```
ipn:0.3.[0-19]
```

An unordered pattern such as ipn:0.3.[10-19,0-4] would be normalized to

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

### B.3. Combined Patterns

This section contains examples of patterns combining items.

#### B.3.1. Any-Scheme Match

This trivial example matches any possible EID. It's text form is:

*:**

   and its CBOR form is:

```
true
```

#### B.3.2. Any-SSP Match

   These two examples match any ipn-scheme EID, either as text scheme
   or integer respectively:

ipn:**

   and

2:**

   and both have a CBOR form of:

```
[2]
```

#### B.3.3. Multiple Scheme Match

   This example combines exact match items for each scheme together in
   one pattern, it will match dtn://node/service and ipn:0.3.4 It's
   text form is:

```
dtn://node/service|ipn:0.3.4
```

   and its CBOR form is:

```
[
  [1, ["node", "service"]],
  [2, [0, 3, 4]]
]
```

## Acknowledgments

   The DTN pattern expressiveness is based on use case examples
   provided by Carlo Caini and Lucien Loiseau.

## 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