NDN Message Format Proposal
draft-stapp-icnrg-ndn-msgs-00

Abstract

This memo defines an on-the-wire format for Named-data Networking (NDN) protocol messages. NDN packets begin with a fixed-size header. After the header, the remainder of the message is composed of type-length-value (TLV) tuples. The Type and Length fields use a compact, fixed-size encoding scheme. The TLVs for message attributes that are standardized are defined in a registry. Part of the TLV number-space is reserved for standardized attributes. Other parts of the number-space are set aside for experimental use.

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 http://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 July 12, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must...
Table of Contents

1. Introduction ............................................. 3
   1.1. Requirements Language .............................. 4
2. NDN Packets and Messages .................................. 4
   2.1. TLV Encoding ...................................... 6
   2.2. Packet Header Options .............................. 7
   2.3. NDN Names .......................................... 8
   2.4. Vendor-specific TLVs .............................. 9
   2.5. Examples ........................................... 9
3. Interest Packets ........................................... 9
   3.1. Message Header ..................................... 10
   3.2. TLV Section ....................................... 10
4. Data Packets .............................................. 10
   4.1. Message Header ..................................... 11
   4.2. TLV Section ....................................... 11
   4.3. Data Signatures .................................... 11
5. Nack Packets ............................................... 11
   5.1. Packet Header ....................................... 13
   5.2. TLV Section ....................................... 13
6. NDN TLVs .................................................. 13
   6.1. NDN Name ........................................... 13
   6.2. ContentData ....................................... 14
   6.3. Certificate ........................................ 14
   6.4. PublisherPublicKeyDigest .......................... 14
   6.5. PublisherPublicKeyName ............................ 14
   6.6. ContentDigest ..................................... 14
   6.7. PublicKey .......................................... 15
   6.8. KeyName ............................................ 15
   6.9. Signature .......................................... 15
   6.10. SignatureBits ..................................... 15
   6.11. Timestamp ......................................... 15
   6.12. Witness ........................................... 15
   6.13. DigestAlgorithm ................................... 16
   6.14. ContentExpiration ................................ 16
   6.15. CacheTTL .......................................... 16
   6.16. TLV Type Codes ................................... 16
7. Assigned Numbers ......................................... 16
   7.1. Message Header Number Assignments .................. 17
   7.2. Flag Bit Assignments ................................ 17
   7.3. Error Code Assignments ............................ 17
1. Introduction

Named-data Networking (NDN) [TODO cite Van], one of a family of Information-centric Networking protocols, proposes the replacement of IP’s endpoint-to-endpoint communication model with one centered around named objects. In NDN networks, a client issues an Interest packet which names an object. No destination address is present: the network routes the Interest based on the name it carries. When a device - a server, a cache engine, or a router with built-in storage - determines that it can satisfy the Interest, it replies with a Data packet containing a Content object, and a cryptographic signature. No source address is present in any message - the network maintains sufficient state to route a Data packet back to the client. This model gives an NDN network a number of attractive properties, and NDN approaches have attracted interest from the networking research community over the last couple of years.

NDN researchers have proposed a wide range of infrastructure and network services. This memo attempts to describe a somewhat limited baseline that can form the basis for more complex and sophisticated services. The baseline is composed of the fundamental packet transport definition, a set of mandatory-to-implement TLVs, and a definition of the basic security mechanisms.

The work required to process each NDN packet at each network node (router) differs significantly from IP datagram handling. Each NDN router must examine the variable-length name in order to make forwarding decisions. Each NDN router must maintain state in order to be able to correlate Data packets from ‘upstream’ with ‘downstream’ clients; other data in packets may also need to be processed in order to support this requirement. Some routers may verify the signatures on some or all Data messages, as a means of detecting spoofing attempts. Some routers may cache Content objects; the network infrastructure itself becomes capable of acting as a distributed cache for popular or valuable data. The obligation to be stateful offers routers the opportunity to take an active role in congestion control. For all of these reasons, an efficient, clear NDN message encoding is central to the success of the overall NDN architecture.
An NDN packet begins with a small header area; the body of the message is mainly composed of TLV tuples. The Name TLV always appears first in the message, so it’s easy to locate in any message. All TLVs contain a length field, so routers and applications can navigate through messages efficiently. The TLV type-code number-space is divided into several ranges: a range for standardized attributes, a range for vendor-specific use, and a range for experimental use.

This is still very much a work-in-progress. There are plenty of TODOs, including:

- I’ve tried to develop a minimal set of TLVs and behavior: have I left out something crucial to a baseline client?
- Details of content signature computation and verification
- I’ve avoided discussion of other control or hop-by-hop messages (if any), OAM, neighbor discovery/adjacency establishment, etc.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. NDN Packets and Messages

At present, there are two NDN message types: the Interest message, and the Data message. On the wire, an NDN message is carried in an NDN packet, which includes a fixed-size packet header. There are three types of NDN packets: Interest packets, Data packets and Nack packets. Each packet type is discussed in detail below. All NDN packets share several properties:

- Packets begin with a fixed-size header, including a protocol version, the header size, and the size of the entire message
- Optional header type-length-value (TLV) tuples are available for information designed for hop-by-hop processing
- After the packet header and header options, the message begins; the message is itself a TLV
- The Name TLV always appears first within the message TLV
- TLVs may contain sub-TLVs
Each TLV includes the length of the entire TLV, even if sub-TLVs are present.

The NDN Packet Header:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ver      |   pkt type    |          packet-size          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    hop lim    |     flags     |  error/resvd  |     hlen      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            |                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Where:

ver          The protocol version. The current value is 1.

pkt type     The packet type. The current packet types are Interest (1), Content (2), Nack (3).

packet-size  The number of octets in the packet, encoded as a 16-bit integer in network byte-order.

hop lim      A hop limit field, available for loop detection.

flags        Header bitflags field.

error/resvd  Reserved field; used for error return code in Nack packets.

hlen         The number of octets in the header area, including any header option TLVs.

All NDN packets begin with a protocol version and a packet-type. A standards action is REQUIRED in order to increment the protocol version or to assign a new packet-type. Routers MUST discard packets with protocol versions they do not support. In general, we expect that nodes may be able to process packets with some small range of protocol versions. NDN nodes that do not understand how to process a packet based on its version must discard the packet.

The NDN packet size is represented as a 16-bit integer in network byte-order. At this point in the evolution of the NDN protocol, this offers a reasonable range of sizes while allowing for a compact
fixed-size representation. The packet-size field includes the size of the header and all of the packet TLVs.

A one-octet hop-limit field is available for loop detection. Routers MUST decrement the value by one before forwarding a message, and MUST NOT forward a message whose hop limit has reached zero. The initial hop limit value is still a topic of active discussion. We expect that common-sense values like 64 or 128 will be recommended eventually. Other loop detection schemes (such a Nonce value) may be available through use of header options.

The flags field contains single-bit flags for use during hop-by-hop processing.

TODO - define flags here, or below with assigned numbers?

The reserved field is used to carry an error or status code when an Interest packet is returned as a Nack packet. The field is reserved in other packet types at this time.

The header-length field contains the size of the entire header area. This is essentially the offset of the Message TLV in the packet, since the Message TLV always begins immediately after the packet header area. The header-length permits use of variable-sized and optional header option TLVs for a variety of purposes. Header options are described below.

### 2.1. TLV Encoding

We propose a simple NDN TLV encoding that uses 2 octets for Type code and 2 octets for TLV Length.

The TLV Type number space and initial assignments are specified in Section 7.4.
The TLV Length field contains the number of octets in the Value field. This implies that a Length of zero may be valid for some TLV Types. The specification of a TLV may include restrictions or validation rules for the TLV’s Length field.

DISCUSSION:

We feel that this encoding offers a reasonable balance between efficiency and flexibility. More complicated variable-length schemes introduce additional validation and processing complexity, as well as canonicalization requirements. Are there use-cases where this simple encoding is inadequate?

2.2. Packet Header Options

Header options offer information beyond that available in the fixed header area. The header option facility supports experimentation with various hop-by-hop processing extensions during this period of NDN evolution. Header options are encoded as TLVs. The header option type-codes use a dedicated number space, distinct from the TLVs in the message body.

We propose several possible header option TLVs, for use with alternative loop-detection methods and for Quality-of-Service (QOS) marking. These seem to be viable candidates for experimentation with hop-by-hop processing.

Nonce option example:

```
 0                   1                   2                   3
+---------------------+---------------------+---------------------+---------------------+
|                     | 'Nonce'             |                     |
|                     | 12                  |                     |
+---------------------+---------------------+---------------------+
```
DiffServ option example:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              'DSCP'           |                5              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   DSCP bits   |  
+-+-+-+-+-+-+-+-+-+-+-+
```

Flow-label option example:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             'Flow'            |               8               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.3. NDN Names

Every NDN message, unsurprisingly, contains a Name TLV: the Name is required for processing at each NDN node. NDN Names can contain several types of information:

- Information needed to route an Interest towards a publisher
- Information about a specific object, such as a file name
- A segment or ‘chunk’ number, used when a client retrieves part of a large data object
- Other standardized data that may be used by client/host stacks, or during network processing.
- Application-specific or session-specific data that is meaningful to applications running at the client and publisher.

NDN Names contain sub-TLVs, which demarcate the boundaries between these various constituent components. Each ‘label’ in a publisher name and each element in a filename is represented as a NameComponent TLV, for example. The segment number is represented in a NameSegment
TLV. By convention, the NameSegment appears last in the collection of Name sub-TLVs.

DISCUSSION:

Is it really necessary to continue to force all of the information in Interests into the Name? Wouldn’t it be clearer to use the Name only for publisher/routing info, object name info, and segment/sequence number? An InterestData or InterestAppData TLV could hold any other data: session information, encrypted fields, application-specific parameters that are not used in forwarding or caching. There would be benefits there for the routers, who wouldn’t need to process data in the "Name" that are entirely unrelated to the forwarding or caching functions. Currently, the routers incur processing cost for both Interest and Content messages when large Name fields are present. App session data wouldn’t often be cacheable in any case: it would likely be encrypted with a session key. The Name would still need to be unique, but that could be accomplished with a single NameComponent containing ... a timestamp and a random number, or a (reasonably strong) hash value.

2.4. Vendor-specific TLVs

A vendor or industry group can introduce its own NDN TLV space using the vendor-specific encapsulation TLV with type code NDN_TLV_VendorSpecific (Section 7.4). The VendorSpecific TLV MUST contain a vendor identifier in the NDN_TLV_VendorId (Section 7.4) TLV. Any other contents of the container TLV are determined by the organization identified. Typically these will be further TLVs - that’s our recommendation – but that is not mandatory. The VendorId must be unique and controlled by the organization using this mechanism. Example VendorIds include an OID, or a DNS name. The VendorId TLV must occur first in the VendorSpecific contents: implementations have to be able to decode the identifier in order to determine whether they can proceed and decode the remainder of the TLV’s data.

2.5. Examples

TODO -- add a couple of encoding examples.

3. Interest Packets
The NDN Interest packet is specified as:

```
INTEREST-PACKET := PACKET-HEADER
                   INTEREST-TLV
                   NAME-TLV [ NDN-TLV ... ]
```

```
PACKET-HEADER := Common packet header area defined above, with
                packet-type == Interest
```

```
INTEREST-TLV := NDN-TLV with Type == InterestMessage
```

```
NAME-TLV := NDN-TLV with Type == Name
```

### 3.1. Message Header

TODO -- brief overview; describe any special header options

### 3.2. TLV Section

The Name TLV is always the first TLV inside the message TLV; this allows network nodes to find the Name easily. Another TLV that is processed at network nodes is the InterestLifetime. Note that we have specified the units used in the InterestLifetime as milliseconds. Nodes should validate any InterestLifetime value presented by a client: we’d expect that routers would have maximum and minimum lifetime values that would bound the lifetime values in Interest messages.

**DISCUSSION:**

We propose an Interest message simpler than the CCNx 0.x scheme. The selectors such as Min- and MaxSuffixComponents, Exclude bloom filter, ChildSelector, etc. are all ... excluded. Routers perform prefix-based matching on Names as they examine their FIBs, and exact-match in their PITs and CSes. Applications that want to offer complex query syntax or filtering need to use application-specific data for those purposes. See Section 2.3 for some discussion about arbitrary application data in Interests.

### 4. Data Packets
The NDN Data packet is specified as:

```
DATA-PACKET := PACKET-HEADER
  DATA-TLV
  NAME-TLV [ NDN-TLV ... ]
```

```
PACKET-HEADER := Common packet header area defined above, with
  packet-type == Data
```

```
DATA-TLV := NDN-TLV with Type == DataMessage
```

```
NAME-TLV := NDN-TLV with Type == Name
```

4.1. Message Header

TODO -- describe any header options etc.

4.2. TLV Section

The Name TLV must be present, and must be the first child of the Data message TLV.

The Data message contains an opaque blob of data in the ContentData TLV accompanied by a cryptographic signature. We propose continuing to use the CCNx approach here, with a few adjustments:

- We allocate new TLV type codes for name information used in the KeyLocator’s KeyName, rather than reusing the message-level Name TLVs.

- The Timestamp is specified to use units of milliseconds.

- For multi-segment content objects, the FinalSegmentID must be present in every Content message.

4.3. Data Signatures

TODO -- Each Data message contains a digital signature. This section describes the process used to produce an NDN signature, encode it in a Data packet payload, and verify it at another node.

5. Nack Packets
The NDN Nack packet is specified as:

\[
\text{NACK-PACKET} := \text{PACKET-HEADER} \\
\text{INTEREST-TLV} \\
\text{NAME-TLV} \ [ \text{NDN-TLV} \ldots ]
\]

\[
\text{PACKET-HEADER} := \text{Common packet header area defined above, with} \\
\text{packet-type} == \text{Nack}
\]

\[
\text{INTEREST-TLV} := \text{NDN-TLV with Type == InterestMessage}
\]

\[
\text{NAME-TLV} := \text{NDN-TLV with Type == Name}
\]

We believe that error-signalling offers value for NDN routers and for NDN end clients. We only specify a mechanism for errors resulting from routers’ processing of Interest messages. NDN routers must maintain state about Interests they have forwarded, and they may be able to take advantage of that state to aid in congestion management and traffic engineering. When a node determines not to forward an Interest, it MAY generate an error message. It is possible to transform the original Interest packet into an error packet in a very direct way: the receiving node changes the packet-type to Nack, sets the packet header error code field to indicate the specific error encountered, and transmits the packet out the face on which the Interest arrived.

This error message offers a basic level of value to other routers and, potentially, to the original client application or protocol stack. Routers on the return path are able to clear their PIT state, without waiting for a timeout. Routers may be able to attempt to re-route the original Interest if they have alternative forwarding paths available. A router can re-produce the original Interest simply clearing the error information from the header and re-transmitting it. A router that does not re-route an Interest after receiving an error may propagate the error by forwarding the Nack packet out the face associated with its entry for the Interest in its own PIT. TODO: should that ‘may’ be a ‘should’?

As the NDN routers propagate a Nack packet, it can make its way along the path from the node encountering an error back to the original client. The basic error message helps the client manage its own sense of the state of the network, identifying congestion issues or NDN names experiencing reachability problems.

The error response to an Interest can also include a hint about the name prefix where the error condition was detected. We expect that the NDN router detecting an error may have a more-specific FIB entry
than routers farther away. An NDN router should include a header option containing the number of name components in its FIB entry that most-closely matched the Interest’s name. This hint may allow a router along the path the error will traverse back to the client to attempt to re-route the Interest if it has an alternative path. The name-prefix hint may also be useful to the original client stack (if the error propagates that far).

NDN routers have to be careful not to allow malicious clients to inject fabricated error messages. An NDN Router must only accept or propagate error messages that arrive via faces associated with neighboring routers, and must not process or propagate an error message that does not match an entry in its PIT.

DISCUSSION:

There are some concerns about introducing a new Error or Nack message-type; some might feel that that threatens the NDN architectural principle permitting only Interest and Data messages. But ... it seems clearer than overloading either of the existing message types, so I’d like to offer it for discussion.

5.1. Packet Header

The Nack packet header includes a field that carries a specific error code. Some

5.2. TLV Section

The Nack message echoes back the Interest TLV message that provoked an error.

6. NDN TLVs

TODO -- add a nice tidy table with info about which T codes can or can’t appear in Interest vs Content messages.

‘Freshness’ vs. ‘ContentExpiration’: one is an interval, one an explicit wall-clock time.

TODO better spec of signing procedure -- ‘SignedFlags’ -- allow the sender to include flags that can be signed?

6.1. NDN Name

The NDN Name is represented as a series of NameComponent TLVs inside a Name TLV. The contents of each NameComponent are opaque octets.
Large data items are transmitted using multiple Interest,Content exchanges for a single name. The NameSegment TLV carries an integer that is appended to the name to identify a specific sub-section of the data. The NameSegment data is a four-byte integer in network byte order. If a NameSegment sub-TLV is present, it must be the last sub-TLV in the Name TLV. More than one NameSegment sub-TLV must not be present in a Name TLV.

TODO -- are four bytes enough: might there be very large files that are presented as very many 1KB or 4KB or 8KB messages? we could a) add a bigger TLV later, b) add a second ‘generation’ field that expands the range of the segment tlv, or c) specify the segment tlv to be eight bytes from the start.

6.2. ContentData

The data in a Content message is represented as an opaque blob in a ContentData TLV. The data is application-specific.

6.3. Certificate

Seems like something that should be supported – ccnx (0.6x) just bails. We’ll need this...

6.4. PublisherPublicKeyDigest

The PublisherPublicKeyDigest identifies the content publisher that signed the content data. The value is a SHA-256 digest of the public key of the publisher. This TLV must be present in each Content message. Its use in signatures and verification is described in Section 4.3.

6.5. PublisherPublicKeyName

The PublisherPublicKeyName identifies the content publisher that signed the content data. The value is an NDN Name that can be used to retrieve the publisher key used to sign a Content object. Its use in signatures and verification is described in Section 4.3.

6.6. ContentDigest

The ContentDigest TLV is optional in Content messages. If present, it contains the SHA-256 hash of the contents of the ContentData TLV, including the TLV Type and Length.
6.7. PublicKey

The PublicKey TLV is optional in Content messages. If present, it contains a public key corresponding to the private key used to sign the ContentData. Its use in signatures and verification is described in Section 4.3.

6.8. KeyName

The KeyName TLV is optional in Content messages. If present, it contains an NDN name that can be used to retrieve keying information that can in turn be used to verify the signature of the ContentData. If present, the KeyName TLV only contains KeyNameComponent sub-TLVs. At least one KeyNameComponent TLV must be present in a KeyName TLV. Its use in signatures and verification is described in Section 4.3.

6.9. Signature

The Signature TLV is a container for other TLVs that carry actual content signature data. The use of a container TLV also allows NDN nodes to skip past a group of related signature data TLVs efficiently if the node is not validating a particular signature. The container may also allow multiple signatures to be present, which may be useful in key agility or crypto algorithm agility. See Section 4.3 for details about content signatures.

6.10. SignatureBits

The SignatureBits TLV contains the actual digital signature in a Content message. See Section 4.3 for details about content signatures.

6.11. Timestamp

TODO -- what’s the signed timestamp really for - what does it add? it gets stuck into the 'ccn_btree_content_payload' struct, but not used? and no one in ccnx seems to use the DTAG_Timestamp in their templates, so...

6.12. Witness

TODO -- should we allocate the code-point for Merkle info, explicitly? then any future "extra" info would also get explicit types...
6.13. DigestAlgorithm

TODO -- default to sha-256. should we allocate more values?

6.14. ContentExpiration

TODO -- prefer to have an absolute expiration time available, seems like it would be useful in some cases?

6.15. CacheTTL

TODO -- ccnx’s "freshness" acts like a cache TTL, seems worth keeping, maybe we should just ... give it a clearer name? and prefer less eccentric units: is seconds good enough? do we need millisecond resolution (really?)

6.16. TLV Type Codes

enum ndn_tlv_e {
    NDN_TLV_Name = 1,
    NDN_TLV_NameComponent = 1,
    NDN_TLV_NameSegment = 2,
    NDN_TLV_ContentData = 4,
    NDN_TLV_Certificate = 5,
    NDN_TLV_SignedInfo = 6,
    NDN_TLV_ContentDigest = 7,
    NDN_TLV_PublicKey = 8,
    NDN_TLV_KeyName = 9,
    NDN_TLV_KeyNameComponent = 1,
    NDN_TLV_Signature = 10,
    NDN_TLV_Timestamp = 11,
    NDN_TLV_Witness = 12,
    NDN_TLV_SignatureBits = 13,
    NDN_TLV_DigestAlgorithm = 14,
    NDN_TLV_FinalSegmentID = 15,
    NDN_TLV_PublisherPublicKeyDigest = 16,
    NDN_TLV_PublisherPublicKeyName = 17,
    NDN_TLV_ContentExpiration = 18,
    NDN_TLV_CacheTTL = 19,
    NDN_TLV_VendorSpecific = 20,
    NDN_TLV_VendorId = 21
};

7. Assigned Numbers
7.1. Message Header Number Assignments

The protocol Version is 1.

The Packet Types are Interest (1), Content (2), and Nack (3).

7.2. Flag Bit Assignments

One flag is defined for the Flags field in the Content message: NoCache (0x01). This flag is a hint to routers and caches that the message is not suitable for caching (a real-time interactive exchange for example.)

7.3. Error Code Assignments

Several Error-code values are defined for the Nack message: NOROUTE (1), CONGESTED (2), NORESOURCE (3), POLICY (4). The NOROUTE error indicates that an NDN node was unable to forward the original Interest because it did not have a route to a publisher matching the message’s Name. The CONGEST error indicates that a node did not forward the Interest because it detected congestion on the link associated with the FIB entry that matched the Name. The NORESOURCE error indicates that the node was unable to forward the Interest because of a shortage of internal resources. The POLICY error indicates that local administrative policy prevented further processing of the Interest.

7.4. TLV Number Ranges

The standardized Type-code number space is partitioned into three regions:

1. Type code zero is reserved and must not be used.

2. Standardized values from 1 to 0x7fff

3. Experimental values from 0x8000 to 0xffff

The vendor-specific TLV mechanism in Section 2.4 allows implementations additional flexibility.

8. The CCNx Scheme

The CCNx project at PARC [TODO citation] has made a significant investment in an on-the-wire scheme, from which the community has learned a great deal. Subsequent experimentation has identified a number of drawbacks to the CCNx scheme that have led us to reconsider some of their choices. We aren’t making an exhaustive analysis of
the CCNx scheme here, but it seems useful to list some of the key points.

- Inconsistent location of the NDN Name
- Ambiguous use of Name-components (within the Name) for multiple purposes
- Inefficient "versioning" and "meta-data" protocol extensions
- Inefficient double-layer encoding
- Confusion between data needed to operate the NDN protocol and application-specific functions
- Multiple encodings for integers, timestamps, etc.
- Reliance on specific byte-values within TLV values for critical protocol control functions
- Issues producing error messages in response to Interests

9. Acknowledgements

TODO -- acknowledge some folks here...

10. IANA Considerations

All drafts must have an IANA section. This memo includes no requests to IANA.

11. Security Considerations

TODO -- point back to signature section?

12. Normative References


Author’s Address