# Identifiers

DPN packages will need globally unique identifiers. These identifiers can either...

1. take a common format, effectively deriving from a common UUID minting procedure, or
2. be an aggregation of a First Node assigned object identifier and the DPN assigned prefix for that First Node
3. creating by using two fields in the registry, one for a standardRFC 4122UUID, and the other for a Node generated UUID. Field 1:RFC 4122 UUID. Field 2: Node UUID. Registry entries map via a standards based UUID to the Node/domain specific UUID and each site can use which ever UUID best suits their needs.

If option #1 is chosen, we will need to decide on a common minting procedure, and implement a federated UUID minting process.
One such option could potentially be CDL's NOID.

If option #2 is chosen, we will need to decide on a common set of guidelines by which all DPN identifiers will conform.

## Package Identifier Pros and Cons

### Option 1

Pros:

- Allows, encourages local/separate minting process
- If following RFC 4122 provides for the most robust solution over the very long term
- Completely Opaque Identifier following RFC 4122, no semantics to ignore
- UUID's will not colide with other UUIDs from other systems
- UUID's will not change during a successioning event
- No prefix necessary
- If using NOIDs, and using common template, can meet uniqueness and robustness criteria, and support prefixes
- Utilises standards, useful for Trac
- Easy to implement

Cons:

- Higher cost of entry for new nodes
- May require mapping between opaque identifiers and local repositories proprietary mechanism
- Original identifiers from a First Node that may someday be needed could be lost or difficult to retrieve

### Option 2

Pros:

- Identity minting/management is under First Node control
- Guarantees uniqueness in the same way Option 1 does: by relying on First Node to do its job
- Original identifiers from a First Node that may someday be needed are preserved
- Lower cost of entry for new nodes
- Supports versioning and succession per the statement below.
- Sufficient for Trac
- Easy to implement

Cons:

- Prefixes required for this option may become semantically intertwined with implementations
- Not standard
- May still require mapping for repositories.
- Identifiers may not be world usable

### Option 3

Pros:

- Satisfies the key elements of both Options 1 and 2
- the local ID -> DPN ID mapping is preserved at the DPN level
- Add protection against long term identifier collision
- Reduces semantics if opaque identifier used
- Allows to version node specific identifier as necessary, or during a succession event.
- Easy to implement

Cons:

- Demands as much work from new nodes as Option 1.
- Adds one extra field in registry, but it is small, 36 bytes (for delimited octets)

**Option 3 has been selected.**

## Impacts on versioning and succession

The general notion here is that whichever option is chosen there is no impact or implication for versioning, succession, or versioning after succession. The idea is that UUIDs are considered opaque regardless of if they are truly opaque (e.g. RFC 4122) or theoretically opaque (e.g. node prefix with UID). In either case, the assumption is that there is no semantic connection between an item's UUID and potentially a previous **version** of that item.

Additionally, independent of which UUID option is chosen, the UUIDs of items that have undergone a **succession** event will not change, even though their first node will change. Finally, the UUIDs of subsequent **versions of successioned items** will also be opaque and conform to the new first node's minting policy.

| | Option 1 | Option 2 | Option 3 |
|---|---|---|---|
| Impacts on versioning | nil | None, per the above statement. | |
| Impacts on successioning | nil | None, per the above statement. | |
| Impacts on versioning after a succession event | nil | None, per the above statement. | Nil for RFC 4122, and allows us to version node specific identifiers |

## DPN Identifier Guidelines

### Character Set

⚠ DPN identifiers will consist of <fill in the blank> characters

1. ASCII ?, with exclusions, such as (i.e. " ":space, "*":aster, etc.)?
2. UTF-8 ?, with exclusions?
3. UNICODE

### Length

⚠ DPN identifiers will not exceed <fill in the blank> number of bytes

1. 128
2. 256
3. 512
4. 1024

### Node Prefix

⚠ DPN identifiers will be prefixed with an opaque, numeric, Node code between 0-999 followed by a colon.
Examples of such codes could be:

- 0:
- 50:
- 123:

## Node specific Unique Identifier and Standard RFC 4122 Universal Unique Identifier

Option 3 entails creating an object identifier by using two fields in the registry:

Field 1: standard RFC 4122 UUID.

Field 2: a Node generated UUID.

These two fields support an identifier mechanism that is both world UUID immutable/opaque and mappable to Node specific identifier mechanisms. This supports a completely transparent Node succession event with identifiers that will not change, and identifiers that may need new versions. This also preserves the mapping of DPN UUID to node specific UUID as part of the distributed registry.

**Field 1:** RFC 4122 UUID.

http://www.ietf.org/rfc/rfc4122.txt

RFC 4122 defines a Uniform Resource Name namespace for UUIDs (Universally Unique IDentifier), also known as GUIDs (Globally Unique IDentifier).  A UUID is 128 bits long, and requires no central registration process. Generating RFC 4122 compliant UUID's is relatively easy and is supported in multiple languages via standard libraries.

For DPN we will use a human readable form using octets.

```
hexOctet = hexDigit hexDigit
      hexDigit =
            "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9" /
            "a" / "b" / "c" / "d" / "e" / "f" /
            "A" / "B" / "C" / "D" / "E" / "F"

A pure Identifier looks like this:

f81d4fae-7dec-11d0-a765-00a0c91e6bf6
```

We may consider using a prefix to indicate that UUIDs are resources.  For example: ***urn:uuid:***

This UUID is completely opaque and world usable, i.e. should not have any collisions with other RFC 4122 generated UUID's. This has an advantage for DPN as DPN UUID's are expected to last at least 50-100 years (or at least till there is a technology refresh). These UUIDs also survive Node succession events and can be used for discovery and administration functions  across all Nodes and Registries. Additionally, the standard format is easily parsed and is small in size, within DPN and without. As new Nodes are added, standard approaches can be used within local registries to manipulate and utilize these UUIDs. Their is a very low likelihood of UUID collision.

**Field 2:** Node UUID.

Each Node may use its own internal UUID in Field 2. This has several advantages for the Node as it may be used in the same manner following the Nodes current practices. We expect the format of this UUID to be different across the Nodes within the DPN consortium, and that after a succession event the UUID may or may not change, or be versioned. The primary advantage is for the convenience of the First Node and will reduce its administrative burden with regards to the archiving of content as the Node UUID will support the Nodes current model for naming and allow for better preservation functions.

Registry entries map via a standards based UUID to the Node/domain specific UUID and each site can use which ever UUID best suits their needs internally, or within the DPN consortium.