# old ARK Identifiers FAQ, version 0.9



# Basics

### How can I give feedback on this document?

By inserting comments into this comment-friendly version.

### What are ARKs?

ARKs (Archival Resource Keys) are high-functioning *identifiers* that lead you to things and to descriptions of those things. For example, this ARK,

`https://n2t.net/ark:/67531/metadc107835/`

gets you to a dissertation, and adding a '?' on the end of the ARK should get you to its description:

`https://n2t.net/ark:/67531/metadc107835/?`

## What's an identifier?

On the internet, an *identifier* is a URL, or part of a URL. For example, this core ARK identifier,

**`ark:/12148/btv1b8449691v/f29`**

appears inside two different URLs (Uniform Resource Locators, also known as web links or web addresses):

`https://gallica.bnf.fr/`**`ark:/12148/btv1b8449691v/f29`**

`https://n2t.net/`**`ark:/12148/btv1b8449691v/f29`**

ARKs are especially good at being *persistent identifiers*.

### What's a persistent identifier?

The average lifetime of a URL was once said to be 44 days. At the end of its life, a URL link *breaks*, meaning it gives you the dreaded "404 Not Found" error that most of us have seen. Irritating as that may be, it's politically awkward when looking for publicly funded research, and it's a cultural disaster for libraries, archives, museums, and other memory organizations.

A *persistent identifier* is a link that in principle keeps working far into the future, even as things move between websites. Normally when things move, everyone who ever recorded the old links would need to be told what the new links are, which is next to impossible. That's where identifier *resolvers* come in.

### What's a resolver?

A *resolver* is a website that specializes in forwarding incoming identifiers (the ones originally advertised to users) to whichever websites are currently best able to deal with them. Overall, forwarding is called *resolution*; one step in a resolution process is called *redirection*.

For a resolver to work, its hostname must be carefully chosen so it won't ever need to be changed. Memory organizations, some of them centuries old, tend to have hostnames well-suited to be resolvers. Some well-known, younger resolvers are n2t.net (the ARK resolver), identifiers.org, doi.org, handle.net, and purl.org.

### What are ARKs used for?

For anything and everything. Current uses of ARKs include

- digital content, such as genealogical records (FamilySearch)
- publisher content (Portico)
- digitized manuscripts (Gallica)
- texts (Internet Archive)
- museum holdings (Smithsonian)
- vocabulary terms (yamz.net, perio.do)
- historical figures (snaccooperative.org)
- datasets, journals, living beings, and more.

## Getting started

### How do I start assigning ARKs?

The only prerequisite is to fill out an online request for a NAAN on behalf of your organization. There is no charge to obtain a NAAN and all memory organizations are welcome. Within a day or two you should receive an email containing a NAAN for your organization's exclusive use. Meanwhile consider the following.

- What things do you want to name with ARKs? Generally you name objects that you own, control, or manage.
- Will you assign ARKs to things contained in larger things that have ARKs? This (granularity) is not a problem, and the '/' character may help.
- Where do you want your ARKs to resolve to? Examples: formatted file, surrogate for a physical thing, landing page with choices, etc.
- Which web server will host your objects? You are asked this when you request a NAAN, even if it's not yet working.
- Which web server/resolver will you use as hostname in the ARK-based URLs that you advertise/publish?

### Are there tools and services to help with ARKs?

There's a partial list of software tools for persistent identification that includes

- Noid (Nice Opaque Identifiers), open source software for minting and resolving ARKs on your own
- ARK plugin for Omeka, which creates and manages ARKs for the Omeka open source web-publishing platform
- ARK module for Drupal, which allows your Drupal site to act as a Name Mapping Authority (NMA)

There are also some vendors, such as ezid.cdlib.org, and some more information on concepts and best practices.

### How do I start serving my ARKs?

It's like serving ordinary URLs. Incoming URL strings get mapped to content that you return, and if your resolver redirects ARKs to those URLs, you're all set. If you're dealing directly with incoming ARK strings, you can map (convert) them to a form your server handles (eg, map them to URLs on arrival). In this second case, your server is acting as a *local resolver*.

If you choose to run your own ARK infrastructure, you get complete autonomy at the expense of maintaining a server/resolver. On the one hand, you might run all custom infrastructure – including content management, web hosting, *minting* (generating unique identifier strings), and running your own server /resolver. That infrastructure could be very simple, such as server configured to map incoming ARK-based URLs to server file pathnames. When you request your NAAN you will be asked to supply the base URL of your local server or resolver.

At the other extreme, you might work with a vendor that supplies all the infrastructure so that, for example, you can focus on creating content. Hybrid solutions are also common, such as just taking your current web server arrangement and just adding an identifier management piece (eg, the API/UI provided by ezid.cdlib.org, which partners with n2t.net).

You will also want to think about whether to advertise (release, publish, disseminate) your ARKs based at your resolver or at n2t.net. You might choose the former for branding or the latter for stability. Resolving your ARKs through n2t.net is always possible, regardless of how you advertise them (this is a side-effect of obtaining a NAAN).

### What is a NAAN, and can I make changes to it?

NAAN stands for Name Assigning Authority Number, which is a unique 5-digit number that begins (after the "ark:" label) every ARK. The NAAN identifies the organization that assigned the ARK and ensures that two different organizations can never assign the same ARK. If your NAAN were 12345 and your resolver were my.example.org, your ARKs would all start with

```
https://my.example.org/ark:/12345/...
```

You may request a NAAN by filling out an an online form. The NAAN you obtain will be listed alongside all other NAANs in the public NAAN registry, which you are free to browse through. Use that same form to update your registry entry, for example, if you make a change to the URL of your resolver, or if you have negotiated with another organization to carry on your work and take over your NAAN. If you transition into or out of a vendor relationship, there is no problem taking your NAAN with you.

NAANs subdivide the set of all possible ARKs (the ARK *namespace*). The subset of ARKs under a given NAAN can be further subdivided into *shoulders* (eg, 12345/x2, 98765/b4), which can make it easy to delegate autonomous ARK assignment to departments in a large organization. ARK resolution is loosely based on NAANs, but because organizations split, ARKs accommodate the namespace splitting problem by supporting management of a namespace by more than one organization.

# ARKs and other identifiers

## Why would I use ARKs compared to, for example, DOIs?

- To keep costs down.
- To work with exactly the metadata you want.
- To be able to create identifiers without metadata.
- To have an identifier as soon as you create the first draft of your data.
- To hold that identifier private while the data and metadata evolve, and decide (maybe years) later, to publish or discard it.
- To retain that identifier upon publication, perhaps then assigning an additional identifier, such as a DOI.
- Because ARKs were built for generic application and don't have to be tortured into identifying physical samples or field stations.
- To be able to change vendor and/or infrastructure without having to coordinate a database transfers with a central authority.
- To be able to deal with the namespace splitting problem without losing control of your identifiers.
- To link identifiers to different kinds of nuanced persistence commitments.
- To be able to add queries (eg, ?lang=en) when resolving your identifiers.
- To use open infrastructure consistent with your organization's values.
- To link directly to the objects you value instead of to landing pages.
- To create one identifier that enables millions (suffix passthrough).
- To access convenient, full-function metadata via inflections.

## What does ARK have in common with DOI, Handle, PURL, and URN?

These are the major persistent identifier types (or schemes). They have all been around since 2001 and they have much in common, starting with structure.

```
 https://n2t.net/ark:/99999/12345

   https://doi.org/10.99999/12345

https://handle.net/10.99999/12345

     https://purl.org/99999/12345

https://<various>/urn:99999:12345
```

ARKs, DOIs, and Handles are all found in places like the Data Citation Index and ORCID.org profiles. As seen in these examples, they all have three parts:

1. the protocol (`https://`) plus a hostname,
2. just for ARK and URN, there's also a label ("ark:" or "urn:"),
3. the name assigning authority (`99999`, `10.99999`, or `purl.org/99999`), which is the organization that created a particular identifier,
4. and finally, the *name*, or local identifier, that it assigned (`12345`).

And they all have little effect on persistence. See 10 persistent myths about persistent identifiers.

## Wait, are you saying ARK, DOI, Handle, PURL, and URN are useless?

No, that's too strong a statement. But let's keep these identifier schemes (types) in perspective.

- They all fail to stop the major causes of broken links: loss of funding, natural disaster, social upheaval, war, deliberate removal, human error, and provider neglect.
- They all require you, the end provider, to update forwarding tables as URLs change.
- They all identify content that is subject to change or removal on future visits.
- They all have identifiers that break regularly and in large numbers – many thousands and more.
- They all give access to almost any kind of thing, whether digital, physical, abstract, person, group, etc.
- A non-trivial fraction of each scheme's identifiers did, and will, fail permanently, requiring forwarding to "tombstone" pages.
- They all rely on ordinary redirection built in to web servers since 1994 and provided for free by hundreds of URL shortening services.

Given how little the schemes do for you, when choosing one you'll likely want to consider factors such as cost, risk, and openness.

## How do ARKs differ from identifiers like DOIs, Handles, PURLs, and URNs?

The short answer is that ARKs are the only mainstream, non-siloed, non-paywalled identifiers that you can register to use in about 48 hours. DOIs, Handles, and PURLs require resolution and other services to come from their respective centralized systems (silos).

That's not to say that persistence is free. Making any identifier persistent burdens you, the provider, with the costs of content management, hosting, monitoring, and forwarding. You can do those things yourself or with help from a vendor. But with ARKs, just as with URLs, you will not be charged separately for your identifiers and you will not be locked in to a special-purpose resolution silo that also locks out other identifiers.

ARKs are unusual in being decentralized. While one *can* get resolution services from a global ARK resolver called n2t.net, over 90% of the ARKs in the world are published without reference to it. More than 500 registered organizations across the world have created an estimated 3.2 billion ARKs, and, as with URLs, no one has ever paid an identifier fee to create them. Of course *maintaining* them isn't free. It is never without cost to keep content access persistent in the long term, regardless of identifier type.

## How else do the identifier types differ?

Here are some more differences between ARKs, DOIs, Handles, PURLs, and URNs.

- Landing pages: Crossref and DataCite DOIs link to publisher landing pages constructed around but *not directly to* objects you care about, but ARKs can freely link *directly to* objects you care about, which is machine- and human-friendly since it does not require an extra human navigation step for common tasks such as
    - opening an article's PDF file for reading,
    - referencing an image file meant to be incorporated automatically inline into a document, and
    - citing a spreadsheet to be used for direct data analysis software.
- DOIs, Handles, etc. do not support ARK-style inflections that permit access to metadata regardless of whether an identifier points to an object or its landing page.
- Unlike DOIs and Handles, ARKs don't have metadata requirements. ARKs that haven't been released into the world are easy to delete.
- All things eventually pass, including hostnames and the web itself and the "`https://`" protocol. When that first part of the identifier ceases to have meaning, only ARKs and URNs will include the label (eg, "ark:") indicating the type of identifier that remains.
- For DOIs, Handles, and PURLs, you are required to use their respective resolvers. ARKs and URNs, permit you to use your own resolver.
- To create DOIs and Handles, you are required to pay a membership fee and, for DOIs, per-DOI charges. There are no fees for ARKs, PURLs, and URNs.
- To create Handles, you are required to install and maintain a local Handle server, which gives you another system to monitor, patch, and troubleshoot.
- Although you can use a local or vendor resolver for your ARKs and URNs, ARKs can be resolved via the global n2t.net resolver.
- The envisioned URN resolution infrastructure was never built, so URNs are currently resolved as URLs, and there is no designated global URN-as-URL resolver. In order to register to create URNs, you must apply for a URN namespace.

ARKs have some unique features that support early object development: ARKs can be deleted, can be born with no metadata, and can exist with any metadata you care to store.

## But if ARKs can be deleted, how can they be trusted?

Being able to delete identifiers actually makes ARKs more trustworthy. The ability to delete is a vital part of healthy collection management that is denied to those non-ARK identifier types prohibiting deletion under the presumption that people, once they are asked to commit, won't make mistakes.

People armed with software regularly turn simple human errors into big tangles of systematic mistakes, even at the threshold of commitment. By making it difficult to clean them up, we force systems to drag those messes forward in perpetuity.

While not immune to such mistakes, ARKs have the big advantage that they can be created and deleted in the shadows, independent of release, publication, or archival commitment.

## Can an object have both an ARK and a DOI?

Yes. Sometimes having two identifiers is useful, although it can become confusing when it happens often. Many people start by assigning ARKs to each thing they create in order to have a stable reference right from the beginning, even before they know whether they want to publish it, let alone keep it. Starting with an ARK, you benefit from being able to keep the original identifier from birth through to public release as the object and its metadata matures. For the subset of things that you end up wanting to publish in places that require DOIs, you can assign DOIs at publication time. This is a way in which ARKs support early object development.

In such a scenario, to reduce the burden of maintaining both identifiers you could register the DOI to redirect to the ARK. At the cost of maintaining just one identifier (the ARK), this would keep newly published links and links previously stored and bookmarked by your collaborators from breaking.

## When should I use ARKs compared to DOIs, Handles, PURLs, or URNs?

There are no simple answers. Identifiers (not things, but their names) are tricky to talk about, so if you hear simple answers elsewhere, beware of common fallacies.

Nothing inherent in ARKs, DOIs, Handles, PURLs, or URNs makes them more or less fit for any particular field, domain, or sector. With an identifier resolver and administrative management service, they all provide the core service of resolution (and so do properly managed URLs).

Generalizations about identifier types sometimes apply when resolution and management for that type is locked into one particular vendor or provider. For example, many PURL and Handle features and restrictions are well-defined by their respective administration silos, as are those of DOIs, which are built on top of Handles. But DOIs have metadata practices that are diverse and evolving across different DOI registration agencies.

The concrete differences that we experience, such as *metadata*, landing pages, and tool integration (eg, publishing tools), are not properties of identifier schemes per se, but properties of resolution, management, and citation services that various providers extend to or withhold from different identifier types. Those services are shaped in turn by communities of practice and by markets. Basic services are founded on a reliable database storing each identifier along with metadata elements (creator, title, date, redirection URL, etc) that describe the identified object. Extra services include link checking, duplicate detection, report generation, and searching.

### What are usage trends for ARKs, DOIs, Handles, PURLs, and URNs?

As of 2019, purely on an incomplete and anecdotal level, here are a few trends that have been observed.

- ARKs have seen broad adoption in cultural memory institutions – museums, archives, and libraries. There is strong adoption in France and francophone regions.
- DOIs until recently have mostly been known as reliable identifiers for scientific and scholarly literature, when in fact this applies to a subset of DOIs assigned via Crossref. What it means to be a DOI is becoming harder to pin down because DOIs are being assigned to datasets, data management plans, field stations, etc. via DataCite, as well as to movies (eg, "Kung Fu Panda") via EIDR. Having said that, Crossref and DataCite DOIs have been successful in creating tools and services for scholarly publishers.
- PURLs have seen lots of use in identifying metadata vocabulary and ontology terms.

### I've heard of ORCIDs, RORs, and UUIDs – where do they fit in?

Those are special kinds of persistent identifiers. ORCIDs (Open Researcher and Contributor Identifiers) only identify researchers, and they link to research works using ARKs, DOIs, etc. ORCIDs look like

```
https://orcid.org/0000-0001-7604-8041
```

ROR (Research Organization Registry) identifiers designate organizations. For example, here's the California Digital Library:

```
https://ror.org/03yrm5c26
```

UUIDs are globally unique, 37-character strings that are easy for software to generate but only become usable as web addresses when made part of a URL, for example, in this ARK:

```
https://somehost.example.com/3c2e39526-e0c3-41ae-be4f-07558a9458eb
```

While embedding a UUID in an ordinary URL makes it *actionable* ("clickable"), you could expect more if it were embedded in an ARK such as

```
https://n2t.net/ark:/65665/3c2e39526-e0c3-41ae-be4f-07558a9458eb
```

As an ARK, for example, that UUID should return metadata (if available) and be insensitive to the hyphens, making this form equally viable:

```
https://n2t.net/ark:/65665/3c2e39526e0c341aebe4f07558a9458eb
```

# From cradle to grave

## When in my workflow should I create ARKs?

At object birth, or even before. We name our babies before they're born, and we name and refer to objects in the conception stages, sometimes long before they bear fruit. Depending on how elaborate the planning may be, your unborn objects could have full-function ARKs that resolve to an appropriate surrogate and return rich metadata, including persistence statements.

The only caveat is to be careful releasing (advertising) ARKs that have uncertain long-term prospects. Some identifier management systems have features to help manage and resolve unreleased identifiers (eg, EZID has a "reserved" status). The more people who know about an ARK, the harder it is to delete.

## How is it that ARKs can be easy to delete?

If no one knows about an identifier but you, there's no harm in deleting or withdrawing it. Stepping back, an identifier is actually an assertion that a given string of characters is associated with specific thing. The fewer people you tell, the easier it is to scrap that assertion. If you create a URL and share it only with your closest colleagues, that is much easier to withdraw than if the URL appeared for a month on a public website, from which it was harvested by internet search engines. In contrast, it is hard to delete DOIs and Handles because once registered and made resolvable, they are effectively released to the world.

ARKs behave like URLs in this respect. Providers are free to create and share ARKs narrowly, in which case they're easy to delete.

Perhaps surprisingly, even if shared more broadly, ARKs can come with persistence statements that tell you how much or how little commitment is made to them. ARKs were designed to articulate a variety of persistence statements, but they are certainly not alone among identifiers and objects that exhibit a variety of commitment "flavors". This is why ARKs are more accurately known as high-functioning rather than persistent identifiers.

Finally, people make mistakes. ARKs, DOIs, Handles, PURLs, and URNs are sometimes broadcast in error and need to be withdrawn. When that happens, provider best practice is make the withdrawn identifier resolve to a page that explains and perhaps apologizes for the inconvenience. Despite the rumors, persistent identifiers are never guaranteed.

### What is meant by ARKs supporting early object development?

People need identifiers before they know exactly what object they refer to, or if they refer to anything worth keeping. An identifier that requires mature metadata cannot be created during early development since little is known about the object. So object creators almost always initially assign identifiers that have no metadata requirements, such as URLs or ARKs.

If you start with an ARK, you benefit from being able to keep the original identifier through to public release as the metadata matures. Many objects go through intensive development and revision phases, sometimes lasting years, during which they are too immature to meet most metadata requirements. Nonetheless every object needs some sort of identifier from conception to maturity, where maturity could look like public release and further enhancement, or abandonment. It is easy to abandon ARKs that have not been released into the world.

Like the object itself, metadata *elements* need a flexible place to grow and mature over time:

- starting in the planning phase, when it just needs an *identifier*,
- at the moment of birth, when its first digital representation needs a *redirection target* URL,
- after the first analysis, when its significance and a tentative *title* emerges,
- when creating dozens of discipline-specific metadata elements that violate most metadata standards except your own,
- during post-processing by a colleague whose name you will add as an additional *creator*,
- when early feedback based on the tweeted identifier turns up a key insight and a new *contributor*,
- and so forth, through to archiving, abandonment, public release, correction, revision, enhancement, etc.

Unlike Crossref and DataCite DOIs, which require specific metadata (eg, see the DataCite schema), ARKs do not constrain any of these activities. Moreover the N2T.net resolver actually supports all of them.

## If ARKs don't require it, why bother creating metadata?

Creating metadata (extra information associated with or describing an object) has several key benefits. First, no matter what the ARK redirects to – whether a landing page or a file – metadata gives users vital information about the object, such as references to newer versions, creation date, provenance, etc. For ARKs typically metadata is accessed via inflections.

Metadata also eases some persistence pain. By themselves, persistent identifier strings are often *opaque*, revealing little about what they identify (because non-opaque identifiers do not age or travel well). But opaque identifiers are difficult because they give you no clues as to what the identifiers were meant to identify. In the absence of metadata you are forced to access the object itself to remind yourself what it is, and to trust that it's the correct object. Metadata really helps. Moreover, discrepancies between returned metadata and the accessed object help everyone detect identifier changes and errors.

Metadata is for grownups, and is far less important for immature objects and their identifiers than for those that have been released. Metadata demonstrates basic provider credibility and commitment to high-functioning identifiers. Not every provider is up to this task.

It need not be expensive. Building metadata from scratch can be costly, but it's usually created and managed by object providers, in which case it can be leveraged efficiently for identifiers. Ideally, for strong persistence master metadata (maintained by object providers) should be reflected in independent systems so that it is hard for someone to tamper undetectably with identifier associations. For example, digital object repositories that obtain ARKs and DOIs from the EZID service store a copy of their metadata with EZID.cdlib.org, which in turn stores another copy with the N2T.net resolver.

## What metadata is recommended for ARKs?

Metadata is messy business for all identifiers, not just ARKs. Across domains and object types there are thousands of standards, many of them overlapping yet conflicting, and each is applied according to local organizational customs and with varying levels of compliance. Choosing or creating a specification for your metadata depends on factors such as

- whether you are currently managing metadata (*hint*: stay with it unless you have a good reason to switch),
- whether you want to officially publish objects (*hint*: prepare to be able to supply author, title, date, publisher/archive, and object type),
- the requirements and capabilities of your resolver (*hint*: your IT staff or vendor might have its own requirements), and
- whether you want to store non-standard elements (*hint*: N2T allows this, but most standards and vendors don't).

Reliable cross-domain interoperation may remain out of reach, but Dublin Core, DataCite, Schema.org JSON-LD, and Dublin Kernel are common metadata specifications to consider for use with ARKs.

## What is Dublin Kernel metadata for (who, what, when, where)?

An ARK typically has a four-element kernel of highly generic metadata, followed by any other metadata elements (name/value pairs) the provider wishes to provide. Since 2001 ARKs were meant to be interoperably indexable across the kernel elements for all object types digital, physical, and abstract. This was an unusually generic descriptive goal, shared by the seminal metadata standard, Dublin Core (DC), which nonetheless underlies most non-generic (domain-specific) metadata standards.

Kernel metadata is structured as if in answer to the questions of who, what, when, and where regarding the expression of, or the "telling" of, an object:

- *who* "told" it (similar to DC Creator, Contributor, and Publisher, but also Inventor, Discoverer, Conductor, etc),
- *what* the "telling" was called (similar to DC Title, but also TissueSampleNumber, ArtifactBarcode, etc),
- *when* it was "told" (similar DC Date, but includes date ranges, approximate and BCE dates),
- *where* the "telling" can be found (from DC Identifier, but usually not needed because this is the ARK itself)

There's much more to say about ARK metadata (available soon at arks.org) and too much to cover in a basic FAQ. Other elements are key, such as

- *how* it was "told" (similar to ResourceType), which may dictate mappings to external metadata specs and additional elements
- redirection target URL, which is usually stored as a distinguished element of metadata
- elements holding persistence statements, to express the strength or weakness of an archival commitment

## What is an ARK "inflection" and how does it differ from "content negotiation"?

An *inflection* is a change to the ending of a word to express a shift in meaning. It permits us to define a word such as "go" without also defining "goes" and "going". To an ARK that leads to an object, simply adding a '?' to the end (an example of an ARK inflection) permits us to request metadata without having to define a separate identifier for the object's metadata. This simple technique can be used by a human with a web browser. The N2T resolver supports both inflections and content negotiation.

*Content negotiation for metadata* is a kludgy software technique for requesting alternate formats of an object, such as the PDF or RTF form of an HTML file. Although not designed for it, historic "content negotiation" was contorted in certain contexts to request metadata under the startling assumption that formats often used to hold metadata *are* in fact metadata and will never be objects in their own right. Unlike inflections, "content negotiation for metadata" doesn't work at all for *objects* represented in those formats (the list of which is growing and known only by private agreement), nor is it easy enough to be used directly by most human users.

Although inflections are commonly associated with ARKs, they are not "owned" by ARKs. Contrary to popular belief, identifiers don't *do* anything – it's their resolvers that *do* or *don't* support such features. So, for example, inflections and suffix passthrough are supported by n2t.net for all identifier types, but not by doi.org or handle.net for any identifier types.

# Resolvers

## If most ARKs run on their own resolvers, why is there also a global resolver for ARKs?

Most ARKs are created by organizations that advertise ("publish") them based at their own resolvers. For example, this ARK was published based at the gallica.bnf.fr resolver:

`https://gallica.bnf.fr/`**`ark:/12148/btv1b8449691v/f29`**

Having to run and maintain your own resolver is the cost of complete autonomy. Using your own resolver also lets you do branding via the hostname, the downside being that brands are transient and tend to make identifiers fragile. Political and even legal (eg, trademarks) pressures may make supporting older branded hostnames, hence their identifiers, difficult.

That's another reason for having the global ARK resolver. People coming across a broken identifier in the future may find its hostname no longer exists, and if it's an ARK they can extract the core identity (starting with "ark:") and present it to the global n2t.net resolver, as in

`https://n2t.net/`**`ark:/12148/btv1b8449691v/f29`**

To avoid the risk of future inconvenience, an organization – even one that runs its own resolver – may choose from the outset to advertise ("publish") its ARKs based at n2t.net.

## Why does the global ARK resolver (n2t.net) *not* have the word "ARK" in it?

When demand for a global ARK resolver arose, basic principles of openness and generality prevented the designers from creating yet another silo in the DOI/Handle/PURL mold. Instead, the ARK resolver was built to be a generic, scheme-agnostic resolver called N2T (Name-to-Thing), which now resolves over 600 types of identifier, including ARKs, DOIs, Handles, PURLs, URNs, ORCIDs, ISSNs, etc. Resolution is essentially looking in a table for an identifier string, regardless of type, and redirecting it to the right place.

The same basic principles guided the design of an earlier tool called noid, which was built for ARKs but is also regularly used by organizations that mint Handles.

## What do you mean by silos?

Typically, scheme-based services are designed as silos, or closed platforms, serving a particular identifier type such as Handle, DOI, or PURL. Each silo performs the same main functions – mapping names (identifiers strings) to things (objects or metadata). Excluding all but one type of identifier string may help to capture markets, but it's wasteful and non-inclusive. It requires building the same set of services over and over for each type and violates basic principles of openness.

In contrast the N2T (Name-to-Thing) resolver and EZID (identifiers made easy) management interface were designed to work with all identifiers. Effort put into any new feature can be efficiently leveraged across all types, which sometimes creates surprising flexibility. For example, ARKs are often stored in EZID with "DOI metadata", and every DOI stored in N2T can benefit from "ARK resolution features" such as inflections and suffix passthrough, which are not available via the main DOI resolver (doi.org).