Network Working Group
Internet-Draft
Intended status: Informational
Expires: 9 May 2026

J. Kunze
Drexel University
E. Bermティs
テ営ole nationale des Chartes

5 November 2025

The ARK Identifier Scheme
draft-kunze-ark-42

Abstract

The ARK (Archival Resource Key) naming scheme is designed to facilitate the high-quality and persistent identification of information objects. The label "ark:" marks the start of a core ARK identifier that can be made actionable by prepending the beginning of a URL. Meant to be usable after today's networking technologies become obsolete, that core should be recognizable in the future as a globally unique ARK independent of the URL hostname, HTTP, etc. A founding principle of ARKs is that persistence is purely a matter of service and neither inherent in an object nor conferred on it by a particular naming syntax. The best any identifier can do is lead users to services that support robust reference. A full-functioning ARK leads the user to the identified object and, with the "?info" inflection appended, returns a metadata record and a commitment statement that is both human- and machine-readable. Tools exist for minting, binding, and resolving ARKs.

Responsibility for this Document

The ARK Alliance Technical Working Group [ARKAtech] is responsible for the content of this Internet Draft. The group homepage lists monthly meeting notes and agendas starting from March 2019. Revisions of the spec are maintained on github at [ARKdrafts].

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://arks-org.github.io/arkspec/draft-ark-spec.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-kunze-ark/.

Source for this draft and an issue tracker can be found at https://github.com/arks-org/arkspec.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes a scheme for the high-quality naming of information resources. The scheme, called the Archival Resource Key (ARK), is well suited to long-term access and identification of any information resources that accommodate reasonably regular electronic description. This includes digital documents, databases, software, and websites, as well as physical objects (books, bones, statues, etc.) and intangible objects (chemicals, diseases, vocabulary terms, performances). Hereafter the term "object" refers to an information resource. The term ARK itself refers both to the scheme and to any single identifier that conforms to it. A reasonably concise and accessible overview and rationale for the scheme is available at [ARK].

Schemes for persistent identification of network-accessible objects are not new. In the early 1990's, the design of the Uniform Resource Name [RFC2141] responded to the observed failure rate of URLs by articulating an indirect, non-hostname-based naming scheme and the need for responsible name management. Meanwhile, promoters of the Digital Object Identifier [DOI] succeeded in building a community of

providers around a mature software system [Handle] that supports name management. The Persistent Uniform Resource Locator [PURL] was another scheme that had the advantage of working with unmodified web browsers. ARKs represent an approach that attempts to build on the strengths and to avoid the weaknesses of these schemes. For example, like URNs, ARKs have an internal label ("ark:") to help them be recognizable as globally unique identifiers in a post-HTTP Internet. Unlike DOIs and Handles, ARKs can be created without centralized fee-based infrastructures. ARK resolvers can take advantage of advanced resolution features such as content negotiation (like DOIs) and suffix passthrough [SPT] (similar to PURL partial redirects). Like PURLs, ARKs openly embrace URLs as the best current choice for actionability.

A founding principle of the ARK is that persistence is purely a matter of service. Persistence is neither inherent in an object nor conferred on it by a particular naming syntax. Nor is the technique of name indirection -- upon which URNs, Handles, DOIs, and PURLs are founded -- of central importance. Name indirection is an ancient and well-understood practice; new mechanisms for it keep appearing and distracting practitioner attention, with the Domain Name System (DNS) [RFC1034] being a particularly dazzling and elegant example. What is often forgotten is that maintenance of an indirection table is an unavoidable cost to the organization providing persistence, and that cost is equivalent across naming schemes. That indirection has always been a native part of the web while being so lightly utilized for the persistence of web-based objects indicates how unsuited most organizations will probably be to the task of table maintenance and to the much more fundamental challenge of keeping the objects themselves viable.

Persistence is achieved through a provider’s successful stewardship of objects and their identifiers. The highest level of persistence will be reinforced by a provider’s robust contingency, redundancy, and succession strategies. It is further safeguarded to the extent that a provider’s mission is shielded from funding and political instabilities. These are by far the major challenges confronting persistence providers, and no identifier scheme has any direct impact on them. In fact, some schemes may actually be liabilities for persistence because they create short- and long-term dependencies for every object access on complex, special-purpose infrastructures, parts of which are proprietary and all of which increase the carry-forward burden for the preservation community. It is for this reason that the ARK scheme relies only on educated name assignment and light use of general-purpose infrastructures that are maintained mostly by the Internet community at large (the DNS, web servers, and web browsers).

As purely a matter of service, persistence is difficult, not known to be commercially attractive, and likely to be undertaken by only a small fraction of content providers that have preservation in their mission. This vision runs counter to some early predictions that technology-backed persistent identifiers would somehow become ubiquitous. On the plus side, persistent identifier solutions should not need to be "internet scale".

1.1. Reasons to Use ARKs

If no persistent identifier scheme contributes directly to persistence, why not just use URLs? A particular URL may be as durable an identifier as it is possible to have, but nothing distinguishes it from an ordinary URL to the recipient who is wondering if it is suitable for long-term reference. An ARK embedded in a URL provides some of the necessary conditions for credible persistence, inviting access to not one, but to three things: to the object, to its metadata, and to a nuanced statement of commitment from the provider in question (the NMA, described below) regarding the object. Existence of the extra service can be probed automatically by appending "?info" to the ARK.

The form of the ARK also supports the natural separation of naming authorities into the original name assigning authority and the diverse multiple name mapping (or servicing) authorities that in succession and in parallel will take over custodial responsibilities from the original assigner (assuming the assigner ever held that responsibility) for the large majority of a long-term object’s archival lifetime. The name mapping authority, indicated by the hostname part of the URL that contains the ARK, serves to launch the ARK into cyberspace. Should it ever fail (and there is no reason why a well-chosen hostname for a 100-year-old cultural memory institution shouldn’t last as long as the DNS), that host name is considered disposeable and replaceable. Again, the form of the ARK helps because it defines exactly how to recover the core immutable object identity, and simple algorithms (one based on the URN model) or even by-hand Internet query can be used for for locating another mapping authority.

There are tools to assist in generating ARKs and other identifiers, such as [NOID] and "uuidgen", both of which rely for uniqueness on human-maintained registries. This document also contains some guidelines and considerations for managing namespaces and choosing hostnames with persistence in mind.

1.2. Three Requirements of ARKs

The first requirement of an ARK is to give users a link from an object to a promise of stewardship for it. That promise is a multi-faceted covenant that binds the word of an identified service provider to a specific set of responsibilities. It is critical for the promise to come from a current provider and almost irrelevant, over a long period of time, what the original assigner's intentions were. No one can tell if successful stewardship will take place because no one can predict the future. Reasonable conjecture, however, may be based on past performance. There must be a way to tie a promise of persistence to a provider's demonstrated or perceived ability -- its reputation -- in that arena. Provider reputations would then rise and fall as promises are observed variously to be kept and broken. This is perhaps the best way we have for gauging the strength of any persistence promise.

The second requirement of an ARK is to give users a link from an object to a description of it. The problem with a naked identifier is that without a description real identification is incomplete. Identifiers common today are relatively opaque, though some contain ad hoc clues reflecting assertions that were briefly true, such as where in a filesystem hierarchy an object lived during a short stay. Possession of both an identifier and an object is some improvement, but positive identification may still be uncertain since the object itself might not include a matching identifier or might not carry evidence obvious enough to reveal its identity without significant research. In either case, what is called for is a record bearing witness to the identifier's association with the object, as supported by a recorded set of object characteristics. This descriptive record is partly an identification "receipt" with which users and archivists can verify an object's identity after brief inspection and a plausible match with recorded characteristics such as title and size.

The final requirement of an ARK is to give users a link to the object itself (or to a copy) if at all possible. Persistent identification plays a vital supporting role but, strictly speaking, it can be construed as no more than a record attesting to the original assignment of a never-reassigned identifier. Object access may not be feasible for various reasons, such as a transient service outage, a catastrophic loss, a licensing agreement that keeps an archive "dark" for a period of years, or when an object's own lack of tangible existence confuses normal concepts of access (e.g., a vocabulary term might be "accessed" through its definition). In such cases the ARK's identification role assumes a much higher profile. But attempts to simplify the persistence problem by decoupling access from identification and concentrating exclusively on the latter are of questionable utility. A perfect system for assigning forever

unique identifiers might be created, but if it did so without reducing access failure rates, no one would be interested. The central issue -- which may be crudely summed up as the "HTTP 404 Not Found" problem -- would not have been addressed.

The central duty of an ARK is a high-quality experience of access and identification. This means supporting reliable access during the period described in its stewardship promise and, failing that, supporting reliable access to a record describing the thing the ARK is associated with.

ARK resolvers must support the "?info" inflection for requesting metadata. Older versions of this specification distinguished between two minimal inflections: ’?’ (brief metadata) and ’??’ (more metadata). While these older inflections are still reserved, because they have proven hard to recognize in some environments, supporting them is optional.

1.3. Organizing Support for ARKs: Our Stuff vs. Their Stuff

An organization and the user community it serves can often be seen to struggle with two different areas of persistent identification: the Our Stuff problem and the Their Stuff problem. In the Our Stuff problem, we in the organization want our own objects to acquire persistent names. Since we possess or control these objects, our organization tackles the Our Stuff problem directly. Whether or not the objects are named by ARKs, our organization is the responsible party, so it can plan for, maintain, and make commitments about the objects.

In the Their Stuff problem, we in the organization want others’ objects to acquire persistent names. These are objects that we do not own or control, but some of which are critically important to us. But because they are beyond our influence as far as support is concerned, creating and maintaining persistent identifiers for Their Stuff is not especially purposeful or feasible for us to engage in. There is little that we can do about someone else’s stuff except encourage their uptake or adoption of persistence services.

Co-location of persistent access and identification services is natural. Any organization that undertakes ongoing support of true persistent identification (which includes description) is well-served if it controls, owns, or otherwise has clear internal access to the identified objects, and this gives it an advantage if it wishes also to support persistent access to outsiders. Conversely, persistent access to outsiders requires orderly internal collection management procedures that include monitoring, acquisition, verification, and change control over objects, which in turn requires object identifiers persistent enough to support auditable record keeping practices.

Although organizing ARK support under one roof thus tends to make sense, object hosting can successfully be separated from name mapping. An example is when a name mapping authority centrally provides uniform resolution services via a protocol gateway on behalf of organizations that host objects behind a variety of access protocols. It is also reasonable to build value-added description services that rely on the underlying services of a set of mapping authorities.

Supporting ARKs is not for every organization. By requiring specific, revealed commitments to preservation, to object access, and to description, the bar for providing ARK services is higher than for some other identifier schemes. On the other hand, it would be hard to grant credence to a persistence promise from an organization that could not muster the minimum ARK services. Not that there isn't a business model for an ARK-like, description-only service built on top of another organization's full complement of ARK services. For example, there might be competition at the description level for abstracting and indexing a body of scientific literature archived in a combination of open and fee-based repositories. The description-only service would have no direct commitment to the objects, but would act as an intermediary, forwarding commitment statements from object hosting services to requestors.

1.4. Definition of Identifier

An identifier is not a string of character data -- an identifier is an association between a string of data and an object. This abstraction is necessary because without it a string is just data. It's nonsense to talk about a string's breaking, or about its being strong, maintained, and authentic. But as a representative of an association, a string can do, metaphorically, the things that we expect of it.

Without regard to whether an object is physical, digital, or conceptual, to identify it is to claim an association between it and a representative string, such as "Jane" or "ISBN 0596000278". What gives a claim credibility is a set of verifiable assertions, or metadata, about the object, such as age, height, title, or number of pages. In other words, the association is made manifest by a record (e.g., a cataloging or other metadata record) that vouches for it.

In the complete absence of any testimony (metadata) regarding an association, a would-be identifier string is a meaningless sequence of characters. To keep an externally visible but otherwise internal string from being perceived as an identifier by outsiders, for example, it suffices for an organization not to disclose the nature of its association. For our immediate purpose, actual existence of an association record is more important than its authenticity or verifiability, which are outside the scope of this specification.

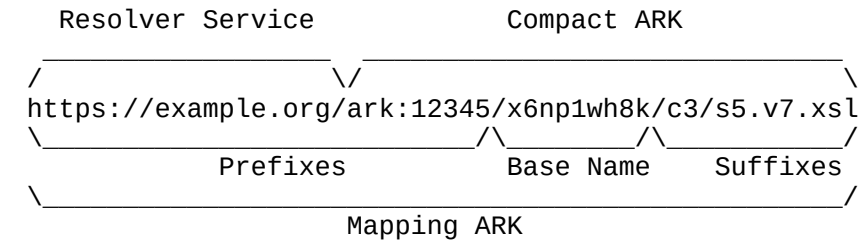
It is a gift to the identification process if an object carries its own name as an inseparable part of itself, such as an identifier imprinted on the first page of a document or embedded in a data structure element of a digital document header. In cases where the object is large, unwieldy, or unavailable (such as when licensing restrictions are in effect), a metadata record that includes the identifier string will usually suffice. That record becomes a conveniently manipulable object surrogate, acting as both an association "receipt" and "declaration".

Note that our definition of identifier extends the one in use for Uniform Resource Identifiers [RFC3986]. The present document still sometimes (ab)uses the terms "ARK" and "identifier" as shorthand for the string part of an identifier, but the context should make the meaning clear.

2. ARK Anatomy

An ARK is represented by a sequence of characters (a string) that contains the Label, "ark:", optionally preceded by the beginning part of a URL. Here is a diagrammed example.

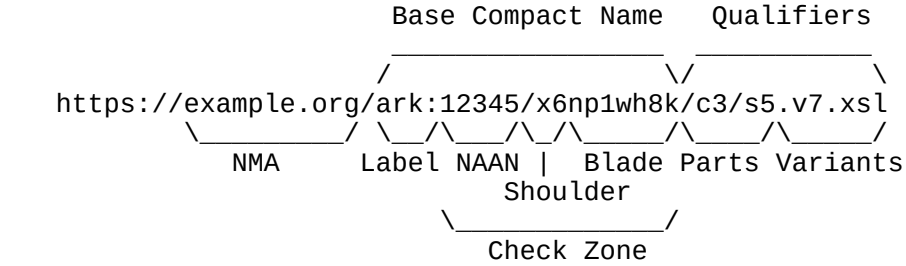
ANATOMY OVERVIEW
==========



When embedded in a URL, an ARK consists of a Compact ARK preceded by a Resolver Service. The larger URL-based ARK is known as a Mapping ARK because it is ready to be mapped (resolved) to an information response (eg, a PDF or metadata). A Mapping ARK is also know as a "fully qualified ARK". The Resolver Service, which need not be limited to URLs in the future, maps the URL according to rules and abilities of an NMA (Name Mapping Authority). The same URL string minus the Resolver Service component is known as a Compact ARK. The Compact ARK is globally unique and may be resolvable via different Resolver Services over time (eg, when one archive succeeds another) or at the same time (eg, when one archive backs up another).

At a high level, after the Label comes the NAAN (Name Assigning Authority Number) followed by the Name that it assigns to the identified thing. The Base Name has Prefixes (NAAN, Label, possibly a Resolver Service) and optional Suffixes to identify Parts and Variant forms. During resolution, a Resolver Service such as n2t.net [N2T] may be able to deal with inflections query strings, and content negotiation.

ANATOMY DETAILS
==========



In a closer view, the Compact ARK consists of a Base Compact Name followed potentially by Qualifiers. The Base Name often, but not necessarily, consists of a Shoulder (for subdividing a NAAN namespace) followed by a Blade. If a check character is present in

an ARK, by convention it is the right-most character of the Base Name, and will have been computed over the string of characters preceding it back to the beginning of the NAAN. This string, including the check character itself, is the Check Zone.

Like the ARK itself, the NAAN "12345" and Shoulder "x6" have compact and fully qualified forms.

Form	Base	Compact Form	Fully Qualified Form
NAAN	12345	ark:12345	https://example.org/ark:12345
Shoulder	x6	ark:12345/x6	https://example.org/ark:12345/x6

Table 1: Example base, compact, and fully qualified form components.

The ARK syntax can be summarized,

[https://NMA/]ark:[/]NAAN/Name[Qualifiers]

where the NMA, '/', and Qualifier parts are in brackets to indicate that they are optional. The Base Compact Name is the substring comprising the "ark:" label, the NAAN and the assigned Name. The Resolver Service is replaceable and makes the ARK actionable for a period of time. Without the Resolver Service part, what remains is the Core Immutable Identity (the "persistible") part of the ARK.

2.1. The Name Mapping Authority (NMA)

Before the "ark:" label may appear an optional Name Mapping Authority (NMA) that is a temporary address where ARK service requests may be sent. Preceded by a URI-type protocol designation such as "https://", it specifies a Resolver Service. The NMA itself is an Internet hostname or host/port combination, optionally followed by URI-type path components, all ending in a '/'. The hostname has the same format and semantics as the host/port part of a URL. In any optional path that follows it, the path is considered to end with the '/' in the first occurrence of "/ark:".

The most important thing about the NMA is that it is "identity inert" from the point of view of object identification. In other words, ARKs that differ only in the optional NMA part identify the same object. Thus, for example, the following three ARKs are synonyms for just one information object:

```
http://example.org/rslvr/ark:12345/x6np1wh8k
     https://example.com/ark:12345/x6np1wh8k
                          ark:12345/x6np1wh8k
```

Strictly speaking, in the realm of digital objects, these ARKs may lead over time to somewhat different or diverging instances of the originally named object. It can be argued that divergence of persistent objects is not desirable, but it is widely believed that digital preservation efforts will inevitably lead to alterations in some original objects (e.g, a format migration in order to preserve the ability to display a document). If any of those objects are held redundantly in more than one organization (a common preservation strategy), chances are small that all holding organizations will perform the same precise transformations and all maintain the same object metadata. More significant divergence would be expected when the holding organizations serve different audiences or compete with each other.

The NMA part makes an ARK into an actionable URL. As with many Internet parameters, it is helpful to approach the NMA being liberal in what you accept and conservative in what you propose. From the recipient’s point of view, the NMA part should be treated as temporary, disposable, and replaceable. From the NMA’s point of view, it should be chosen with the greatest concern for longevity. A carefully chosen NMA should be at least as permanent as the providing organization’s own hostname. In the case of a national or university library, for example, there is no reason why the NMA could not be considerably more permanent than soft-funded proxy hostnames such as hdl.handle.net, dx.doi.org, and purl.org. In general and over time, however, it is not unexpected for an NMA eventually to stop working and require replacement with the NMA of a currently active service provider.

This replacement relies on a mapping authority "resolver" discovery process, of which two alternate methods are outlined in a later section. The ARK, URN, Handle, and DOI schemes all use a resolver discovery model that sooner or later requires matching the original assigning authority with a current provider servicing that authority’s named objects; once found, the resolver at that provider performs what amounts to a redirect to a place where the object is currently held. All the schemes rely on the ongoing functionality of currently mainstream technologies such as the Domain Name System [RFC1034] and web browsers. The Handle and DOI schemes in addition require that the Handle protocol layer and global server grid be available at all times.

The practice of prepending "https://" and an NMA to an ARK is a way of creating an actionable identifier by a method that is itself temporary. Assuming that infrastructure supporting [RFC2616] information retrieval will no longer be available one day, ARKs will then have to be converted into new kinds of actionable identifiers. By that time, if ARKs see widespread use, web browsers would presumably evolve to perform this (currently simple) transformation automatically.

2.2. The ARK Label Part (ark:)

The label part distinguishes an ARK from an ordinary identifier. There is a new form of the label, "ark:", and an old form, "ark:/", both of which must be recognized in perpetuity. Implementations should generate new ARKs in the new form (without the "/") and resolvers must always treat received ARKs as equivalent if they differ only in regard to new form versus old form labels. Thus these two ARKs are equivalent:

```
    ark:/12345/x6np1wh8k
     ark:12345/x6np1wh8k
```

In a URL found in the wild, the label indicates that the URL stands a reasonable chance of being an ARK. If the context warrants, verification that it actually is an ARK can be done by testing it for existence of the three ARK services.

Since nothing about an identifier syntax directly affects persistence, the "ark:" label (like "urn:", "doi:", and "hdl:") cannot tell you whether the identifier is persistent or whether the object is available. It does tell you that the original Name Assigning Authority (NAA) had some sort of hopes for it, but it doesn’t tell you whether that NAA is still in existence, or whether a decade ago it ceased to have any responsibility for providing persistence, or whether it ever had any responsibility beyond naming. An NAA identifies an autonomous assignment stream for a set of objects as well as a reference to help locate a resolver for them. Often, NAA policies and practices reflect an organization (department, project, data center, periodical, etc.) in which it is embedded. An organization may have more than one NAA, for example, a publisher may have a distinct NAA for each of its three journals.

Only a current provider can say for certain what sort of commitment it intends, and the ARK label suggests that you can query the NMA directly to find out exactly what kind of persistence is promised. Even if what is promised is impersistence (i.e., a short-term identifier), saying so is valuable information to the recipient. Thus an ARK is a high-functioning identifier in the sense that it provides access to the object, the metadata, and a commitment statement, even if the commitment is explicitly very weak.

2.3. The Name Assigning Authority Number (NAAN)

Recalling that the general form of the ARK is,

    [https://NMA/]ark:[/]NAAN/Name[Qualifiers]

the part of the ARK directly following the "ark:" (or older "ark:/") label is the Name Assigning Authority Number (NAAN), up to but not including the next '/' (slash) character. This part is always required, as it identifies the organization that originally assigned the Name of the object. Typically the organization is an institution, a department, a laboratory, or any group that conducts a stable, policy-driven name assigning effort. An organization may request a NAAN from the ARK Maintenance Agency [ARKagency] (described in Appendix A) by filling out the form [NAANrequest].

For received ARKs, implementations must support a minimum NAAN length of 16 octets. NAANs are opaque strings of one or more "betanumeric" characters, specifically,

    bcdfghjkmnpqrstvwxz0123456789

which consists of digits and consonants, minus the letter 'l'. Restricting NAANs to betanumerics (alphanumerics without vowels or 'l') serves two goals. It reduces the chances that words -- past, present, and future -- will appear in NAANs and carry unintended semantics. It also helps usability by not mixing commonly confused characters ('0' and 'O', '1' and 'l') and by being compatible with strong transcription error detection (eg, the [NOID] check digit algorithm). Since 2001, every assigned NAAN has consisted of exactly five digits.

The NAAN designates a top-level ARK namespace. Once registered for a namespace, a NAAN is never re-registered. It is possible, however, for there to be a succession of organizations that manage an ARK namespace.

There are currently four NAANs available for assignment on reserved shoulders (see the Shoulder section) by all organizations. An ARK bearing one of these NAANs carries a specific, immutable meaning that recipients can rely on for long term pragmatic benefit as described below.

Shared NAAN meaning	The immutable purpose, meaning, or connotation of ARKs bearing this NAAN.	Expect to resolve?	OK for long term reference?
12345 examples	Example ARKs appearing in documentation. They might resolve, but link checkers usually need not be concerned if they don’t. They should not be considered viable for long term reference.	maybe	no
99152 terms	ARKs for controlled vocabulary and ontology terms, such as metadata element names and pick-list values. They should resolve to term definitions and are suitable for long term reference.	yes	yes
99166 agents	ARKs for people, groups, and institutions as "agents" (actors, such as creators, contributors, publishers, performers, etc). They should resolve to agent definitions and are suitable for long term reference.	yes	yes
99999 test ids	ARKs for test, development, or experimental purposes, often at scale. They might resolve, but link checkers usually need not be concerned if they don’t. They should not be considered viable for long term reference.	maybe	no

Table 2: Four NAANs shared across all ARK-assigning organizations.

To make use of a shared NAAN, an organization has several options described in Section 2.4.1.

2.4. The Name Part

The part of the ARK just after the NAAN is the Name assigned by the NAA, and it is also required. Semantic opaqueness in the Name part is strongly encouraged in order to reduce an ARK's vulnerability to era- and language-specific change. Identifier strings containing linguistic fragments can create support difficulties down the road. No matter how appropriate or even meaningless they are today, such fragments may one day create confusion, give offense, or infringe on a trademark as the semantic environment around us and our communities evolves.

Names that look more or less like numbers avoid common problems that defeat persistence and international acceptance. The use of digits is highly recommended. Mixing in non-vowel alphabetic characters (eg, betanumerics) a couple at a time is a relatively safe and easy way to achieve a denser namespace (more possible names for a given length of the name string). Such names have a chance of aging and traveling well. The absence of recognizable words makes typos harder to detect in opaque strings, so a common mitigation is to add a check character. Tools exists that mint, bind, and resolve opaque identifiers, with or without check characters [NOID]. More on naming considerations is given in a subsequent section.

2.4.1. Optional: Shoulders

Just as an ARK namespace is subdivided by NAANs reserved for NAAs, it is generally advantageous for an NAA to subdivide its own NAAN namespace into "shoulders", where each shoulder is reserved for an internal department or unit. Like the NAAN, which is a string of characters that follows the "ark:" label, a shoulder is a string of characters (starting with a "/") that extends the NAAN. The base compact name assigned by the NAA consists of the NAAN, the shoulder, a final string known as the "blade". (The shoulder plus blade terminology mirrors locksmith jargon describing the information-bearing parts of a key.)

The blade string is chosen by the NAA such that the string created by concatenating the NAAN plus shoulder plus blade becomes the unique base object name. Otherwise the blade may come from any source, for example, it might come from a counter, a timestamp, a [NOID] minter, a legacy 100-year-old accession number, etc. If there is a check digit, it is expected to appear at the end of the blade and to be computed over the base compact name minus the label part (see Check Zone), which is generally the most important part of an ARK to make

opaque. In particular, check digits are not expected to cover qualifiers, which often name subobjects of a persistent object that are less stable and less opaquely named than the parent object (for example, ten years hence, the object's thumbnail image will be of a higher resolution and the OCR text file will be re-derived with improved algorithms.

It is important not to use any delimiter between the shoulder string and blade string, especially not a "/" since it declares an object boundary (see the section on ARKs that reveal object hierarchy).

```
ark:12345/x6np1wh8k/c2/s4.pdf       # correct primordinal shoulder
ark:12345/x6/np1wh8k/c2/s4.pdf      # INCORRECT
            ^ WRONG
```

This little bit of discretion shields organizations from end users making inferences about expected levels of support based on recognizable shoulders. To help in-house ARK administrators reliably know where the shoulder ends, it is recommended to use the "first-digit convention" so that shoulders are "primordinal". A primordinal shoulder is a sequence of one or more betanumeric characters ending in a digit, as shown above. This means that the shoulder is all consonant letters (often just one) after the NAAN and "/" up to and including the first digit encountered after the NAAN. One property of primordinal shoulders is that there is an infinite number of them possible under any NAAN.

To help manage each namespace into the future, NAAs are encouraged to create shoulders, even if there is only one to start with. If an organization wishes to create a shoulder under one of shared NAANs (99999, 12345, 99152, or 99166, described in Table 2), it should fill out the Shoulder Request Form [shoulderrequest].

2.5. The Qualifier Part

The part of the ARK following the NAA-assigned Name is an optional Qualifier. It is a string that extends the Base Name in order to create a kind of service entry point into the object named by the NAA. At the discretion of the providing NMA, such a service entry point permits an ARK to support access to individual hierarchical components and subcomponents of an object, and to variants (versions, languages, formats) of components. A Qualifier may be invented by the NAA or by any NMA servicing the object.

In form, the Qualifier is a ComponentPath, or a VariantPath, or a ComponentPath followed by a VariantPath. A VariantPath is introduced and subdivided by the reserved character '.', and a ComponentPath is introduced and subdivided by the reserved character '/'. In this example,

    https://example.org/ark:12345/x6np1wh8k/c3/s5.v7.xsl

the string "/c3/s5" is a ComponentPath and the string ".v7.xsl" is a VariantPath. The ARK Qualifier is a formalization of some currently mainstream URL syntax conventions. This formalization specifically reserves meanings that permit recipients to make strong inferences about logical sub-object containment and equivalence based only on the form of the received identifiers; there is great efficiency in not having to inspect metadata records to discover such relationships. NMAs are free not to disclose any of these relationships merely by avoiding the reserved characters above. Hierarchical components and variants are discussed further in the next two sections.

The Qualifier, if present, differs from the Name in several important respects. First, a Qualifier may have been assigned either by the NAA or later by the NMA. The assignment of a Qualifier by an NMA effectively amounts to an act of publishing a service entry point within the conceptual object originally named by the NAA. For our purposes, an ARK extended with a Qualifier assigned by an NMA will be called an NMA-qualified ARK.

Second, a Qualifier assignment on the part of an NMA is made in fulfillment of its service obligations and may reflect changing service expectations and technology requirements. NMA-qualified ARKs could therefore be transient, even if the base, unqualified ARK is persistent. For example, it would be reasonable for an NMA to support access to an image object through an actionable ARK that is considered persistent even if the experience of that access changes as linking, labeling, and presentation conventions evolve and as format and security standards are updated. For an image "thumbnail", that NMA could also support an NMA-qualified ARK that is considered impersistent because the thumbnail will be replaced with higher resolution images as network bandwidth and CPU speeds increase. At the same time, for an originally scanned, high-resolution master, the NMA could publish an NMA-qualfied ARK that is itself considered persistent. Of course, the NMA must be able to return its separate commitments to unqualified, NAA-assigned ARKs, to NMA-qualified ARKs, and to any NAA-qualified ARKs that it supports.

A third difference between a Qualifier and a Name concerns the semantic opaqueness constraint. When an NMA-qualified ARK is to be used as a transient service entry point into a persistent object, the priority given to semantic opaqueness observed by the NAA in the Name part may be relaxed by the NMA in the Qualifier part. If service priorities in the Qualifier take precedence over persistence, short-term usability considerations may recommend somewhat semantically laden Qualifier strings.

Finally, not only is the set of Qualifiers supported by an NMA mutable, but different NMAs may support different Qualifier sets for the same NAA-identified object. In this regard the NMAs act independently of each other and of the NAA.

The next two sections describe how ARK syntax may be used to declare, or to avoid declaring, certain kinds of relatedness among qualified ARKs.

2.5.1. ARKs that Reveal Object Hierarchy

An NAA or NMA may choose to reveal the presence of a hierarchical relationship between objects using the '/' (slash) character after the Name part of an ARK. Some authorities will choose not to disclose this information, while others will go ahead and disclose so that manipulators of large sets of ARKs can infer object relationships by simple identifier inspection; for example, this makes it possible for a system to present a collapsed view of a large search result set.

If the ARK contains an internal slash after the NAAN, the piece to its left indicates a containing object. For example, publishing an ARK of the form,

```
ark:12345/x54/xz/321
```

is equivalent to publishing three ARKs,

```
ark:12345/x54/xz/321
ark:12345/x54/xz
ark:12345/x54
```

together with a declaration that the first object is contained in the second object, and that the second object is contained in the third.

Revealing the presence of hierarchy is completely up to the assigner (NMA or NAA). It is hard enough to commit to one object's name, let alone to three objects' names and to a specific, ongoing relatedness among them. Thus, regardless of whether hierarchy was present

initially, the assigner, by not using slashes, reveals no shared inferences about hierarchical or other inter-relatedness in the following ARKs:

ark:12345/x54_xz_321
ark:12345/x54_xz
ark:12345/x54xz321
ark:12345/x54xz
ark:12345/x54

Note that slashes around the ARK's NAAN (/12345/ in these examples) are not part of the ARK's Name and therefore do not indicate the existence of some sort of NAAN super object containing all objects in its namespace. A slash must have at least one non-structural character (one that is neither a slash nor a period) on both sides in order for it to separate recognizable structural components. So initial or final slashes may be removed, and double slashes may be converted into single slashes.

2.5.2. ARKs that Reveal Object Variants

An NAA or NMA may choose to reveal the possible presence of variant objects or object components using the '.' (period) character after the Name part of an ARK. Some authorities will choose not to disclose this information, while others will go ahead and disclose so that manipulators of large sets of ARKs can infer object relationships by simple identifier inspection. This makes it possible for a system to present a collapsed view of a large number of search result items without having to issue database queries in order to retrieve and analyze the inter-relatedness among all of those items.

If the ARK contains an internal period after the Name, the piece to the left of the first such period is a root name and the piece to its right, and up to the end of the ARK or to the next period is a suffix. A Name may have more than one suffix, for example,

ark:12345/x54.24
ark:12345/x4z/x54.24
ark:12345/x54.v18.fr.odf

There are two main rules. First, if two ARKs share the same root name but have different suffixes, the corresponding objects were considered variants of each other (different formats, languages, versions, etc.) by the assigner (NMA or NAA). Thus, the following ARKs are variants of each other:

```
ark:12345/x54.v18.fr.odf
ark:12345/x54.321xz
ark:12345/x54.44
```

Second, publishing an ARK with a suffix implies the existence of at least one variant identified by the ARK without its suffix. The ARK is otherwise silent about what additional variants might exist. So publishing the ARK,

```
ark:12345/x54.v18.fr.odf
```

is equivalent to publishing the four ARKs,

```
ark:12345/x54.v18.fr.odf
ark:12345/x54.v18.fr
ark:12345/x54.v18
ark:12345/x54
```

Revealing the possibility of variants is completely up to the assigner. It is hard enough to commit to one object's name, let alone to multiple variants' names and to a specific, ongoing relatedness among them. The assigner is the sole arbiter of what constitutes a variant within its namespace, and whether to reveal that kind of relatedness by using periods within its names.

A period must have at least one non-structural character (one that is neither a slash nor a period) on both sides in order for it to separate recognizable structural components. So initial or final periods may be removed, and adjacent periods may be converted into a single period.

3. ARK Processing

3.1. Character Repertoires

The Name and Qualifier parts are strings of visible ASCII characters. For received ARKs, implementations must support a minimum length of 255 octets for the string composed of the Base Name plus Qualifier. Implementations generating strings exceeding this length should understand that receiving implementations may not be able to index such ARKs properly. Characters may be letters, digits, or any of these seven characters:

```
=   ~   *   +   @   _   $
```

The following characters may also be used, but their meanings are reserved:

% - . /

The characters ’/’ and ’.’ are ignored if either appears as the last character of an ARK. If used internally, they allow a name assigner to reveal object hierarchy and object variants as previously described.

Hyphens are considered to be insignificant and are always ignored in ARKs. A ’-’ (hyphen) may appear in an ARK for readability, or it may have crept in during the formatting and wrapping of text, but it must be ignored in lexical comparisons. As in a telephone number, hyphens have no meaning in an ARK. It is always safe for an NMA that receives an ARK to remove any hyphens found in it. As a result, like the NMA, hyphens are "identity inert" in comparing ARKs for equivalence. For example, the following ARKs are equivalent for purposes of comparison and ARK service access:

ark:12345/x5-4-xz-321
https://sneezy.dopey.com/ark:12345/x54--xz32-1
ark:12345/x54xz321

The ’%’ character is reserved for %-encoding all other octets that would appear in the ARK string, in the same manner as for URIs [RFC3986]. A %-encoded octet consists of a ’%’ followed by two uppercase hex digits; for example, "%7D" stands in for ’}’. Uppercase hex digits are preferred for compatibility with URI encoding conventions, especially useful when URL-based ARKs are compared for equivalence by ARK-unaware software systems; thus use "%ACT" instead of "%acT". The character ’%’ itself must be represented using "%25". As with URNs, %-encoding permits ARKs to support legacy namespaces (e.g., ISBN, ISSN, SICI) that have less restricted character repertoires [RFC2288].

Implementors should be prepared to normalize some common invalid characters that may be found in ARKs copy pasted from processed text. For example, when pasting an ARK that was broken during line wrapping, a user may inadvertently propagate newlines, spaces, hyphens, and hyphen-like characters (eg, U+2010 to U+2015) that were introduced by the publisher. The normalization strategy is up to the implementor and may include converting hyphen-like characters to hyphens and removing whitespace.

3.2. Normalization and Lexical Equivalence

To determine if two or more ARKs identify the same object, the ARKs are compared for lexical equivalence after first being normalized. Since ARK strings may appear in various forms (e.g., having different NMAs), normalizing them minimizes the chances that comparing two ARK strings for equality will fail unless they actually identify different objects. In a specified-host ARK (one having an NMA), the NMA never participates in such comparisons. Normalization described here serves to define lexical equivalence but does not restrict how implementors normalize ARKs locally for storage.

Normalization of a received ARK for the purpose of octet-by-octet equality comparison with another ARK consists of the following steps.

1. The NMA part (eg, everything from an initial "https://" up to the first occurrence of "/ark:"), if present is removed.

2. Any URI query string is removed (everything from the first literal ’?’ to the end of the string).

3. The first case-insensitive match on "ark:/" or "ark:" is converted to "ark:" (replacing any uppercase letters and removing any terminal ’/’).

4. Any uppercase letters in the NAAN are converted to lowercase.

5. In the string that remains, the two characters following every occurrence of ’%’ are converted to uppercase. The case of all other letters in the ARK string must be preserved.

6. All hyphens are removed. Implementors should be aware that non-ASCII hyphen-like characters (eg, U+2010 to U+2015) may arrive in the place of hyphens and, if they wish, remove them.

7. If normalization is being done as part of a resolution step, and if the end of the remaining string matches a known inflection, the inflection is noted and removed.

8. Structural characters (slash and period) are normalized: initial and final occurrences are removed, and two structural characters in a row (e.g., // or ./) are replaced by the first character, iterating until each occurrence has at least one non-structural character on either side.

The resulting ARK string is now normalized. Comparisons between normalized ARKs are case-sensitive, meaning that uppercase letters are considered different from their lowercase counterparts.

To keep ARK string variation to a minimum, no reserved ARK characters should be %-encoded unless it is deliberately to conceal their reserved meanings. No non-reserved ARK characters should ever be %-encoded. Finally, no %-encoded character should ever appear in an ARK in its decoded form.

3.3. Resolver Chains and Roles

To resolve a Compact ARK (ie, an ARK beginning "ark:") it must initially be promoted to a Mapping ARK so that it becomes actionable. On the web, this means finding a suitable web Resolver Service to prepend to the compact form of the identifier in order to convert it to a URL (cf [CURIE]). (This is more or less true for any type of identifier not already in URL form.)

The identifier’s Resolver Service is the first point of contact in the resolution process (eg, the NMA in a typical URL). It can be seen as the "first resolver" because resolution may involve multiple redirections via a chain of resolvers before a resolution response is returned by the last resolver (the "responder"). The chain is as long as the number of redirections. In particular, when the first resolver is also the last resolver, the chain has zero length. Most ARKs using N2T.net as the first resolver will be redirected to a second resolver listed in the record for a given ARK’s NAAN. For example, an ARK bearing the NAAN 12148 (BnF) and the NMA n2t.net (as its first resolver) could be redirected to a second resolver, ark.bnf.fr. Whether n2t.net or ark.bnf.fr will be the first resolver depends on what NMA appears in the ARK at the time of resolution. Currently, BnF ARKs are published with the BnF’s NMA (ark.bnf.fr), so most BnF ARKs will not start with n2t.net.

Resolution in general can be seen as a multi-stage computation that maps a client identifier to some sort of response. On the web, each resolver in the chain is an HTTP server; even if the "responder" (last resolver) is a proxy server that intiates a non-web sub-resolution process, that is invisible to the original client and out of scope for this discussion. A web resolution response may take on a variety of forms, including the return of a landing page, or a metadata record, or a web-based 404 Not Found message. A given response, as well as the specific chain of resolvers traversed, depends not only on the identifier, but also on such things as the time, location, credentials, and technical platform of the client initiating resolution.

Also, for a given identifier, the "responder" (last resolver) for an object request may be different from the responder for a metadata request. While maintenance of objects and their metadata often takes place within one organization, the responder for object requests may

be different from the responder for metadata requests. To add credibility to a persistence promise, it can be useful to maintain a secondary copy of object metadata at an external and publicly visible resolver. For example, N2T.net was originally designed to store a secondary copy of metadata for many millions of identifiers.

If an ARK Resolver Service receives a request for a NAAN that it knows nothing about, best practice is to redirect the request to N2T.net.

3.4. Finding a Resolver Service

In order to discover if a given host or origin [RFC6454] implements an ARK Resolver Service, it is recommended that it respond to the "/.well-known/ark" URI suffix [RFC8615] as described in the IANA Considerations section.

Given either a Compact ARK (missing an NMA) or an ARK with a non-functioning NMA, in order to derive an actionable identifier (these days, a URL), a Resolver Service must be found. On the web, the Resolver Service consists of a URI scheme and an NMA, where the NMA is a host or host/port combination, optionally followed by URI-type path components, all ending in a '/'. The Resolver Service is expected to respond to basic ARK service requests. An NMA may provide mapping services for more than one NAAN.

Upon encountering an ARK, a user (or client software) determines if it is a Mapping ARK (ie, it is a URL beginning with a Resolver Service). If the Resolver Service is working, this discovery step likely can be skipped assuming the URL correctly identifies a working resolver. If a new Resolver Service needs to be found, the client looks inside the ARK again for the NAAN (Name Assigning Authority Number). Querying a global database, it then uses the NAAN to look up all current Resolver Services that service ARKs issued by the identified NAA. This NAAN-to-NMA resolver discovery method is common (cf URN, Handle, DOI) but does not address the namespace splitting problem, which is when a portion of a NAAN space originally maintained entirely by one NMA is taken on by a second NMA; now the NAAN alone cannot reveal which NMA (resolver) to choose.

The global database is key, and ideally the lookup would be automatic and transparent to the user. For this, the current mainstream method is to use the resolver chain that starts with the Name-to-Thing (N2T) Resolver [N2T] at n2t.net. While the sequence of hops in the redirect chain behind N2T is subject to change (as with any redirection strategy that supports persistent naming), the N2T chain is meant to provide several durable capabilities. N2T is a reliable, lightweight Resolver Service provided by the ARK Alliance primarily to support actionable HTTP-based URLs for as long as HTTP is used.

For example, the N2T chain can return metadata about individual ARK-identified resources, but in 2024 n2t.net itself (as the first resolver in the chain) stopped returning such metadata immediately and instead began delegating them to resolvers further along the chain. Similarly, the N2T chain redirects ARKs to local resolvers on a per-NAAN basis, but in 2024 n2t.net stopped redirecting immediately to those resolvers and instead began redirecting them to a subdomain (under arks.org) that held such per-NAAN knowledge. The N2T chain is designed to anticipate future specialty subresolvers that can provide such things as backup copies of per-resource metadata (for resilience) or per-resource redirection when a NAAN namespace splits and not all ARKs under a given NAAN can be redirected by just one NMA.

The knowledge about where a given NAAN can be resolved is contained in a plain text [NAANregistry] file. As a machine- and human-readable database, it contains explanatory comments that can be directly inspected by users, for example, when manually looking for a Resolver Service. An appendix describes an historical way to discover an NMA based on a simplification of the URN resolver discovery method, itself very similar in principle to the resolver discovery method used by Handles and DOIs. None of these methods does more than what can be done with a very small, consortially maintained web server such as [N2T].

In the interests of long-term persistence, however, ARK mechanisms are first defined in high-level, protocol-independent terms so that mechanisms may evolve and be replaced over time without compromising fundamental service objectives. Either or both specific methods given here may eventually be supplanted by better methods since, by design, the ARK scheme does not depend on a particular method, but only on having some method to locate an active NMA.

At the time of issuance, at least one NMA for an ARK should be prepared to service it. That NMA may or may not be administered by the Name Assigning Authority (NAA) that created it. Consider the following hypothetical example of providing long-term access to a cancer research journal. The publisher wishes to turn a profit and a

national library wishes to preserve the scholarly record. An agreement might be struck whereby the publisher would act as the NAA and the national library would archive the journal issue when it appears, but without providing direct access for the first six months. During the first six months of peak commercial viability, the publisher would retain exclusive delivery rights and would charge access fees. Again, by agreement, both the library and the publisher would act as NMAs, but during that initial period the library would redirect requests for issues less than six months old to the publisher. At the end of the waiting period, the library would then begin servicing requests for issues older than six months by tapping directly into its own archives. Meanwhile, the publisher might routinely redirect incoming requests for older issues to the library. Long-term access is thereby preserved, and so is the commercial incentive to publish content.

Although it will be common for an NAA also to run an NMA service, it is never a requirement. Over time NAAs and NMAs will come and go. One NMA will succeed another, and there might be many NMAs serving the same ARKs simultaneously (e.g., as mirrors or as competitors). There might also be asymmetric but coordinated NMAs as in the library-publisher example above.

4. Naming Considerations

The most important threats faced by persistence providers include such things as funding loss, natural disaster, political and social upheaval, processing faults, and errors in human oversight. There is nothing that an identifer scheme can do about such things. Still, a few observed identifier failures and inconveniences can be traced back to naming practices that we now know to be less than optimal for persistence.

4.1. ARKs and Usability

Because linguistic constructs imperil persistence, for ARKs non-ASCII character support is not a priority. ARKs and URIs share goals of transcribability and transportability within web documents, so characters are required to be visible, non-conflicting with HTML/XML syntax, and not subject to tampering during transmission across common transport gateways.

Any measure that reduces user irritation with an identifier will increase its chances of acceptance, hence survival. Irritation can arise when common user assumptions are not shared by service providers. For example, providers may wish to avoid leading zeroes in an identifier component that looks like a number because users who assume that leading zeroes contribute nothing to that quantity may

omit them during transcription. Also, unless an identifier already employs mixed case letters, users often assume uppercase letters to be equivalent to their lowercase counterparts, in which instance (e.g., a shoulder that employs only one case) a provider may wish to accept incoming ARKs in either uppercase or lowercase. Another common user assumption is that hyphens are lexically insignificant. It is fine to publish ARKs with hyphens in them (e.g., such as the output of UUID/GUID generators), but the uniform treatment of hyphens (and their Unicode equivalents) as insignificant reduces the possibility of identifiers breaking when users omit hyphens or when word processors add them.

4.2. Objects Should Wear Their Identifiers

A valuable technique for provision of persistent objects is to try to arrange for the complete identifier to appear on, with, or near its retrieved object. An object encountered at a moment in time when its discovery context has long since disappeared could then easily be traced back to its metadata, to alternate versions, to updates, etc. This has seen reasonable success, for example, in book publishing and software distribution. An identifier string only has meaning when its association is known, and this a very sure, simple, and low-tech method of reminding everyone exactly what that association is.

4.3. Names are Political, not Technological

If persistence is the goal, a deliberate local strategy for systematic name assignment is crucial. Names must be chosen with great care. Poorly chosen and managed names will devastate any persistence strategy, and they do not discriminate by identifier scheme. Whether a mistakenly re-assigned name is a URN, DOI, PURL, URL, or ARK, the damage -- failed access and confusion -- is not mitigated more in one scheme than in another. Conversely, in-house efforts to manage names responsibly will go much further towards safeguarding persistence than any choice of naming scheme or name resolution technology.

Branding (e.g., at the corporate or departmental level) is important for funding and visibility, but substrings representing brands and organizational names should be given a wide berth except when absolutely necessary in the hostname (the identity-inert) part of the ARK. These substrings are not only unstable because organizations change frequently, but they are also dangerous because successor organizations often have political or legal reasons to actively suppress predecessor names and brands. Any measure that reduces the chances of future political or legal pressure on an identifier will decrease the chances that our descendants will be obliged to deliberately break it.

4.4. Choosing a Hostname or NMA

Hostnames appearing in any identifier meant to be persistent must be chosen with extra care. The tendency in hostname selection has traditionally been to choose a token with recognizable attributes, such as a corporate brand, but that tendency wreaks havoc with persistence that is supposed to outlive brands, corporations, subject classifications, and natural language semantics (e.g., what did the three letters "gay" mean in 1958, 1978, and 1998?). Today's recognized and correct attributes are tomorrow's stale or incorrect attributes. In making hostnames (any names, actually) long-term persistent, it helps to eliminate recognizable attributes to the extent possible. This affects selection of any name based on URLs, including PURLs and the explicitly disposable NMAs.

There is no excuse for a provider that manages its internal names impeccably not to exercise the same care in choosing what could be an exceptionally durable hostname, especially if it would form the prefix for all the provider's URL-based external names. Registering an opaque hostname in the ".org" or ".net" domain would not be a bad start. Another way is to publish your ARKs with an organizational domain name that will be mapped by DNS to an appropriate NMA host. This makes for shorter names with less branding vulnerability.

It is a mistake to think that hostnames are inherently unstable. If you require brand visibility, that may be a fact of life. But things are easier if yours is the brand of long-lived cultural memory institution such as a national or university library or archive. Well-chosen hostnames from organizations that are sheltered from the direct effects of a volatile marketplace can easily provide longer-lived global resolvers than the domain names explicitly or implicitly used as starting points for global resolution by indirection-based persistent identifier schemes. For example, it is hard to imagine circumstances under which the Library of Congress' domain name would disappear sooner than, say, "handle.net".

For smaller libraries, archives, and preservation organizations, there is a natural concern about whether they will be able to keep their web servers and domain names in the face of uncertain funding. One option is to form or join a group of like-minded organizations with the purpose of providing mutual preservation support. The first goal of such a group would be to perpetually rent a hostname on which to establish a web server that simply redirects incoming member organization requests to the appropriate member server; using ARKs, for example, a 150-member group could run a very small server (24x7) that contained nothing more than 150 rewrite rules in its configuration file. Even more helpful would be additional consortial support for a member organization that was unable to continue

providing services and needed to find a successor archival organization. This would be a low-cost, low-tech way to publish ARKs (or URLs) under highly persistent hostnames.

There are no obvious reasons why the organizations registering DNS names, URN Namespaces, and DOI publisher IDs should have among them one that is intrinsically more fallible than the next. Moreover, it is a misconception that the demise of DNS and of HTTP need adversely affect the persistence of URLs. At such a time, certainly URLs from the present day might not then be actionable by our present-day mechanisms, but resolution systems for future non-actionable URLs are no harder to imagine than resolution systems for present-day non-actionable URNs and DOIs. There is no more stable a namespace than one that is dead and frozen, and that would then characterize the space of names bearing the "http://" or "https://" prefix. It is useful to remember that just because hostnames have been carelessly chosen in their brief history does not mean that they are unsuitable in NMAs (and URLs) intended for use in situations demanding the highest level of persistence available in the Internet environment. A well-planned name assignment strategy is everything.

4.5. Assigners of ARKs

A Name Assigning Authority (NAA) is an organization that creates (or delegates creation of) long-term associations between identifiers and information objects. Examples of NAAs include national libraries, national archives, and publishers. An NAA may arrange with an external organization for identifier assignment. The US Library of Congress, for example, allows OCLC (the Online Computer Library Center, a major world cataloger of books) to create associations between Library of Congress call numbers (LCCNs) and the books that OCLC processes. A cataloging record is generated that testifies to each association, and the identifier is included by the publisher, for example, in the front matter of a book.

An NAA does not so much create an identifier as create an association. The NAA first draws an unused identifier string from its namespace, which is the set of all identifiers under its control. It then records the assignment of the identifier to an information object having sundry witnessed characteristics, such as a particular author and modification date. A namespace is usually reserved for an NAA by agreement with recognized community organizations (such as IANA and ISO) that all names containing a particular string be under its control. In the ARK an NAA is represented by the Name Assigning Authority Number (NAAN).

The ARK namespace reserved for an NAA is the set of names bearing its particular NAAN. For example, all strings beginning with "ark:12345/" are under control of the NAA registered under 12345, which might be the National Library of Finland. Because each NAA has a different NAAN, names from one namespace cannot conflict with those from another. Each NAA is free to assign names from its namespace (or delegate assignment) according to its own policies. These policies must be documented in a manner similar to the declarations required for URN Namespace registration [RFC2611].

Organizations can request or update a NAAN by filling out the NAAN Request Form [NAANrequest].

4.6. NAAN Namespace Management

Every NAA should have a namespace management strategy. A classic hierarchical approach is to partition a NAAN namespace into subnamespaces known as "shoulders". As explained in Section 2.4.1, each shoulder is a unique prefix that guarantees non-collision of names in different partitions. This practice is strongly encouraged for all NAAs, especially when subnamespace management and assignment streams will be delegated to departments, units, or projects within an organization. For example, with a NAAN that is assigned to a university and managed by its main library, the library should take care to reserve shoulders (semantically opaque shoulders being preferred) for distinct assignment streams. Prefix-based partition management is typically an important responsibility of the NAA.

This shoulder delegation approach plays out differently in two real-world examples: DNS names and ISBN identifiers. In the former, the hierarchy is deliberately exposed and in the latter it is hidden. Rather than using lexical boundary markers such as the period (’.’) found in domain names, the ISBN uses a publisher prefix but doesn’t disclose where the prefix ends and the publisher’s assigned name begins. This practice of non-disclosure, found in the ISBN and ISSN schemes, is encouraged in assigning ARKs because it reduces the visibility of an assertion that is probably not important now and may become a vulnerability later.

If longevity is the goal, it is important to keep the prefixes free of recognizable semantics; for example, using an acronym representing a project or a department is discouraged. At the same time, you may wish to set aside a subnamespace for testing purposes under a shoulder such as "fk9..." that can serve as a visual clue and reminder to maintenance staff that this "fake" identifier was never published.

There are other measures one can take to avoid user confusion, transcription errors, and the appearance of accidental semantics when creating identifiers. If you are generating identifiers automatically, pure numeric identifiers are likeley to be semantically opaque enough, but it’s probably useful to avoid leading zeroes because some users mistakenly treat them as optional, thinking (arithmetically) that they don’t contribute to the "value" of the identifier.

If you need lots of identifiers and you don’t want them to get too long, you can mix digits with consonants (but avoid vowels since they might accidentally spell words) to get more identifiers without increasing the string length. In this case you may not want more than a two letters in a row because it reduces the chance of generating acronyms. Generator tools such as [NOID] provide support for these sorts of identifiers, and can also add a computed check character as a guarantee against the most common transcription errors. If used, it is recommended that the check character be appended to the original Base Compact Name string (ie, minus the check character), that original string having been the basis for computing the check character.

4.7. Sub-Object Naming

As mentioned previously, semantically opaque identifiers are very useful for long-term naming of abstract objects, however, it may be appropriate to extend these names with less opaque extensions that reference contemporary service entry points (sub-objects) in support of the object. Sub-object extensions beginning with a digit or underscore (’_’) are reserved for the possibilty of developing a future registry of canonical service points (e.g., numeric references to versions, formats, languages, etc).

5. Generic ARK Service Definition

An ARK request’s output is delivered information; examples include the object itself, a policy declaration (e.g., a promise of support), a descriptive metadata record, or an error message. The experience of object delivery is expected to be an evolving mix of information that reflects changing service expectations and technology requirements; contemporary examples include such things as an object summary and component links formatted for human consumption. ARK services must be couched in high-level, protocol-independent terms if persistence is to outlive today’s networking infrastructural assumptions. The high-level ARK service definitions listed below are followed in the next section by a concrete method (one of many possible methods) for delivering these services with today’s technology. Note that some services may be invoked in one operation,

such as when an "?info" inflection returns both a description and a permanence declaration for an object.

5.1. Generic ARK Access Service (access, location)

Returns (a copy of) the object or a redirect to the same, although a sensible object proxy may be substituted. Examples of sensible substitutes include,

* a table of contents instead of a large complex document,

* a home page instead of an entire web site hierarchy,

* a rights clearance challenge before accessing protected data,

* directions for access to an offline object (e.g., a book),

* a description of an intangible object (a disease, an event), or

* an applet acting as "player" for a large multimedia object.

May also return a discriminated list of alternate object locators. If access is denied, returns an explanation of the object’s current (perhaps permanent) inaccessibility.

5.1.1. Generic Policy Service (permanence, naming, etc.)

Returns declarations of policy and support commitments for given ARKs. Declarations are returned in either a structured metadata format or a human readable text format; sometimes one format may serve both purposes. Policy subareas may be addressed in separate requests, but the following areas should be covered: object permanence, object naming, object fragment addressing, and operational service support.

The permanence declaration for an object is a rating defined with respect to an identified permanence provider (guarantor), which will be the NMA. It may include the following aspects.

1. "object availability" -- whether and how access to the object is supported (e.g., online 24x7, or offline only),

2. "identifier validity" -- under what conditions the identifier will be or has been re-assigned,

3. "content invariance" -- under what conditions the content of the object is subject to change, and

4. "change history" -- access to corrections, migrations, and revisions, whether through links to the changed objects themselves or through a document summarizing the change history

One approach to persistence statements, conceived independently from ARKs, can be found at [PStatements], with ongoing work available at [ARKspecs]. An older approach to a permanence rating framework is given in [NLMPerm], which identified the following "permanence levels":

> Not Guaranteed: No commitment has been made to retain this resource. It could become unavailable at any time. Its identifier could be changed.
>
> Permanent: Dynamic Content: A commitment has been made to keep this resource permanently available. Its identifier will always provide access to the resource. Its content could be revised or replaced.
>
> Permanent: Stable Content: A commitment has been made to keep this resource permanently available. Its identifier will always provide access to the resource. Its content is subject only to minor corrections or additions.
>
> Permanent: Unchanging Content: A commitment has been made to keep this resource permanently available. Its identifier will always provide access to the resource. Its content will not change.

Naming policy for an object includes an historical description of the NAA's (and its successor NAA's) policies regarding differentiation of objects. Since it is the NMA that responds to requests for policy statements, it is useful for the NMA to be able to produce or summarize these historical NAA documents. Naming policy may include the following aspects.

1. "similarity" -- (or "unity") the limit, defined by the NAA, to the level of dissimilarity beyond which two similar objects warrant separate identifiers but before which they share one single identifier, and

2. "granularity" -- the limit, defined by the NAA, to the level of object subdivision beyond which sub-objects do not warrant separately assigned identifiers but before which sub-objects are assigned separate identifiers.

Subnaming policy for an object describes the qualifiers that the NMA, in fulfilling its ongoing and evolving service obligations, allows as extensions to an NAA-assigned ARK. To the conceptual object that the

NAA named with an ARK, the NMA may add component access points and derivatives (e.g., format migrations in aid of preservation) in order to provide both basic and value-added services.

Addressing policy for an object includes a description of how, during access, object components (e.g., paragraphs, sections) or views (e.g., image conversions) may or may not be "addressed", in other words, how the NMA permits arguments or parameters to modify the object delivered as the result of an ARK request. If supported, these sorts of operations would provide things like byte-ranged fragment delivery and open-ended format conversions, or any set of possible transformations that would be too numerous to list or to identify with separately assigned ARKs.

Operational service support policy includes a description of general operational aspects of the NMA service, such as after-hours staffing and trouble reporting procedures.

5.1.2. Generic Description Service

Returns a description of the object. Descriptions are returned in a structured metadata format, a human-readable text format, or in one format that serves both purposes (such as human-readable HTML with embedded machine-readable metadata, or perhaps YAML). A description must at a minimum answer the who, what, when, and where questions ("where" being the long-term identifier as opposed to a transient redirect target) concerning an expression of the object. Standalone descriptions should be accompanied by the modification date and source of the description itself. May also return discriminated lists of ARKs that are related to the given ARK.

5.2. Overview of The HTTP URL Mapping Protocol (THUMP)

The HTTP URL Mapping Protocol (THUMP) is a way of taking a key (any identifier) and asking such questions as, what information does this identify and how permanent is it? [THUMP] is in fact one specific method under development for delivering ARK services. The protocol runs over HTTP to exploit the web browser’s current pre-eminence as user interface to the Internet. THUMP is designed so that a person can enter ARK requests directly into the location field of current browser interfaces. Because it runs over HTTP, THUMP can be simulated and tested via keyboard-based interactions [RFC0854].

The asker (a person or client program) starts with an identifier, such as an ARK or a URL. The identifier reveals to the asker (or allows the asker to infer) the Internet host name and port number of a server system that responds to questions. Here, this is just the NMA that is obtained by inspection and possibly lookup based on the

ARK’s NAAN. The asker then sets up an HTTP session with the server system, sends a question via a THUMP request (contained within an HTTP request), receives an answer via a THUMP response (contained within an HTTP response), and closes the session. That concludes the connected portion of the protocol.

A THUMP request is a string of characters beginning with a ’?’ (question mark) that is appended to the identifier string. The resulting string is sent as an argument to HTTP’s GET command. Request strings too long for GET may be sent using HTTP’s POST command. The two most common requests correspond to two degenerate special cases. First, a simple key with no request at all is the same as an ordinary access request. Thus a plain ARK entered into a browser’s location field behaves much like a plain URL, and returns access to the primary identified object, for instance, an HTML document.

The second special case is a minimal ARK description request string consisting of just "?info". For example, entering the string,

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

into the browser’s location field directly precipitates a request for a metadata record describing the object identified by ark:67531/metadc107835. The browser, unaware of THUMP, prepares and sends an HTTP GET request in the same manner as for a URL. THUMP is designed so that the response (indicated by the returned HTTP content type) is normally displayed, whether the output is structured for machine processing (text/plain) or formatted for human consumption (text/html). In addition to "?info", this specification reserves both ’?’ and ’??’ (originally older forms) for future use.

The following example THUMP session assumes metadata being returned by a resolver (as server) to a browser client. Each line has been annotated to include a line number and whether it was the client or server that sent it. Without going into much depth, the session has four pieces separated from each other by blank lines: the client’s piece (lines 1-3), the server’s HTTP/THUMP response headers (4-8), and the body of the server’s response (9-18). The first and last lines (1 and 19) correspond to the client’s steps to start the TCP session and the server’s steps to end it, respectively.

```
 1  C: [opens session]
    C: GET https://n2t.net/ark:67531/metadc107835?info HTTP/1.1
    C:
    S: HTTP/1.1 200 OK
 5  S: Content-Type: text/plain
    S: THUMP-Status: 0.6 200 OK
    S: Link: </ark:67531/metadc107835> rel="describes";
    S:
    S: erc:
10  S: who:   Austin, Larry
    S: what:  A Study of Rhythm in Bach's Orgelbテシchlein
    S: when:  1952
    S: where: https://digital.library.unt.edu/ark:/67531/metadc107835
    S: erc-support:
15  S: who:   University of North Texas Libraries
    S: what:  Permanent: Stable Content:
    S: when:  20081203
    S: where: https://digital.library.unt.edu/ark:/67531/
    S: [closes session]
```

The first two server response lines (4-5) above are typical of HTTP. The next line (6) is peculiar to THUMP, and indicates the THUMP version and a normal return status. The final header line (7) asserts, for the benefit of recipients unfamiliar with ARK inflections, that the response describes the uninflected ARK.

The balance of the response consists of a single metadata record (9-18) that comprises the ARK description service response. The returned record is in the format of an Electronic Resource Citation [ERC], which is discussed in overview in the next section. For now, note that it contains four elements that answer the top priority questions regarding an expression of the object: who played a major role in expressing it, what the expression was called, when it was created, and where the expression may be found (note that "where" is preferably a persistent, citable identifier rather than an unstable URL sometimes mistakenly referred to as a "location"). This quartet of elements comes up again and again in ERCs. Lines 13-17 contain a minimal persistence statement.

Each segment in an ERC tells a different story relating to the object, so although the same four questions (elements) appear in each, the answers depend on the segment's story type. While the first segment tells the story of an expression of the object, the second segment tells the story of the support commitment made to it: who made the commitment, what the nature of the commitment was, when it was made, and where a fuller explanation of the commitment may be found.

5.3. The Electronic Resource Citation (ERC)

An Electronic Resource Citation (or ERC, pronounced e-r-c) [ERC] is a kind of object description that uses Dublin Core Kernel metadata elements [DCKernel]. The ERC with Kernel elements provides a simple, compact, and printable record for holding data associated with an information resource. As originally designed [Kernel], Kernel metadata balances the needs for expressive power, very simple machine processing, and direct human manipulation. The ERC sense of "citation" is not limited to the traditional referencing of a result or information fixed in time on a printed page, but to a more general kind of reference, both backward, to digital material that cannot be known to be fixed in time (true of virtually all online information), and forward, to material that is all the more valuable for improving or evolving over time.

The previous section shows two limited examples of what is fully described elsewhere [ERC]. The rest of this short section provides some of the background and rationale for this record format.

A founding principle of Kernel metadata is that direct human contact with metadata will be a necessary and sufficient condition for the near term rapid development of metadata standards, systems, and services. Thus the machine-processable Kernel elements must only minimally strain people’s ability to read, understand, change, and transmit ERCs without their relying on intermediation with specialized software tools. The basic ERC needs to be succinct, transparent, and trivially parseable by software.

Borrowing from the data structuring format that underlies the successful spread of email and web services, the ERC format uses [ANVL], which is based on email and HTTP headers [RFC2822]. There is a naturalness to ANVL’s label-colon-value format (seen in the previous section) that barely needs explanation to a person beginning to enter ERC metadata.

While ANVL elements are expected at the top level and don’t themselves support hierarchy, the value of an ANVL element may be an arbitrary encoded hierarchy of JSON or XML. Typically, the name of such an ANVL element ends in "json" or "xml", for example, "json" or "geojson". Care should be taken to escape structural characters that appear in element names and values, specifically, line terminators (both newlines ("\n") and carriage returns ("\r")) and, in element names, colons (":").

Besides simplicity of ERC system implementation and data entry mechanics, ERC semantics (what the record and its constituent parts mean) must also be easy to explain. ERC semantics are based on a

reformulation and extension of the Dublin Core [RFC5013] hypothesis, which suggests that the fifteen Dublin Core metadata elements have a key role to play in cross-domain resource description. The ERC design recognizes that the Dublin Core's primary contribution is the international, interdisciplinary consensus that identified fifteen semantic buckets (element categories), regardless of how they are labeled. The ERC then adds a definition for a record and some minimal compliance rules. In pursuing the limits of simplicity, the ERC design combines and relabels some Dublin Core buckets to isolate a tiny kernel (subset) of four elements for basic cross-domain resource description.

For the cross-domain kernel, the ERC uses the four basic elements -- who, what, when, and where -- to pretend that every object in the universe can have a uniform minimal description. Each has a name or other identifier, a locator (a means to access it), some responsible person or party, and a date. It doesn't matter what type of object it is, or whether one plans to read it, interact with it, smoke it, wear it, or navigate it. Of course, this approach is flawed because uniformity of description for some object types requires more semantic contortion and sacrifice than for others. That is why at the beginning of this document, the ARK was said to be suited to objects that accommodate reasonably regular electronic description.

While insisting on uniformity at the most basic level provides powerful cross-domain leverage, the semantic sacrifice is great for many applications. So the ERC also permits a semantically rich and nuanced description to co-exist in a record along with a basic description. In that way both sophisticated and naive recipients of the record can extract the level of meaning from it that best suits their needs and abilities. Key to unlocking the richer description is a controlled vocabulary of ERC record types (not explained in this document) that permit knowledgeable recipients to apply defined sets of additional assumptions to the record.

5.4. Advice to Web Clients

ARKs are envisaged to appear wherever durable object references are planned. Library cataloging records, literature citations, and bibliographies are important examples. In many of these places URLs (Uniform Resource Locators) are currently used, and inside some of those URLs are embedded URNs, Handles, and DOIs. Unfortunately, there's no suggestion of a way to probe for extra services that would build confidence in those identifiers; in other words, there's no way to tell whether any of those identifiers is any better managed than the average URL.

ARKs are also envisaged to appear in hypertext links (where they are not normally shown to users) and in rendered text (displayed or printed). A normal HTML link for which the URL is not displayed looks like this.

```
<a href = "https://example.org/index.htm"> Click Here <a>
```

A URL with an embedded ARK invites access (via "?info") to extra services:

```
<a href = "https://example.org/ark:14697/b12345x"> Click Here <a>
```

Using the [N2T] resolver to provide identifier-scheme-agnostic protection against hostname instability, this ARK could be published as:

```
<a href = "https://n2t.net/ark:14697/b12345x"> Click Here <a>
```

An NAA will typically make known the associations it creates by publishing them in catalogs, actively advertizing them, or simply leaving them on web sites for visitors (e.g., users, indexing spiders) to stumble across in browsing.

5.5. Enhancements and Related Specifications

ARK services, data models, inflections, and applications continue to evolve. Follow-on developments and specifications will be made available from the ARK Maintenance Agency [ARKspecs].

5.6. IANA Considerations

This specification registers "ark" in the "Well-Known URIs" registry as defined by [RFC8615].

URI suffix: ark

Change controller: ARK Alliance [ARKagency]

Specification document: this document (this section)

Status: provisional

If a host has an ARK Resolver Service, best practice is to allow clients to discover its location by supporting the URI path

/.well-known/ark

Accessing this path should return a plain text file containing the ARK Resolver Service URI path ending in '/'. Here is an example access.

```
 1  C: [opens session]
    C: GET https://example.org/.well-known/ark HTTP/1.1
    C:
    S: HTTP/1.1 200 OK
 5  S: Content-Type: text/plain
    S:
    S: /index.php/bulletin/gateway/plugin/pubIdResolver/
    S: [closes session]
```

The service URI path should name the root of an ARK Resolver Service on the host in the sense that appending a Compact ARK to it should create a valid resolution request path. Often the service path is just '/', but this cannot be assumed.

5.7. Security Considerations

The ARK naming scheme poses no direct risk to computers and networks. Implementors of ARK services need to be aware of security issues when querying networks and filesystems for Name Mapping Authority services, and the concomitant risks from spoofing and obtaining incorrect information. These risks are no greater for ARK mapping authority discovery than for other kinds of service discovery. For example, recipients of ARKs with a specified NMA should treat it like a URL and be aware that the identified ARK service may no longer be operational.

Apart from mapping authority discovery, ARK clients and servers subject themselves to all the risks that accompany normal operation of the protocols underlying mapping services (e.g., HTTP). As specializations of such protocols, an ARK service may limit exposure to the usual risks. Indeed, ARK services may enhance a kind of security by helping users identify long-term reliable references to information objects.

6. Informative References

[ANVL] Kunze, J., Kahle, B., Masanes, J., and G. Mohr, "A Name-Value Language", 2005, <https://n2t.net/ark:13030/c7x921j3h>.

[ARK] Kunze, J., "Towards Electronic Persistence Using ARK Identifiers", IWAW/ECDL Annual Workshop Proceedings , 3 August 2003, <https://n2t.net/ark:13030/c7n00zt1z>.

[ARKagency]
ARK Alliance, "ARK Maintenance Agency", 2021, <https://arks.org>.

[ARKAtech] ARK Alliance, "ARK Alliance Technical Working Group", 2022, <https://github.com/arks-org/arks.github.io/wiki/ARKA-Technical-WG-wiki>.

[ARKdrafts]
ARK Alliance, "ARK Drafts Repository", 2022, <https://github.com/arks-org/arkspec>.

[ARKspecs] ARK Alliance, "ARK Maintenance Agency Specifications", 2021, <https://arks.org/specs/>.

[CURIE] W3C, "CURIE Syntax 1.0", December 2010, <https://www.w3.org/TR/2010/NOTE-curie-20101216/>.

[DCKernel] Dublin Core Metadata Initiative, "Kernel Metadata Working Group", 20012008, <https://dublincore.org/groups/kernel/>.

[DOI] I. D. Foundation, "The Digital Object Identifier (DOI) System", February 2001, <https://doi.org/10.1000/203>.

[ERC] Kunze, J. and A. Turner, "Kernel Metadata and Electronic Resource Citations", October 2007, <https://n2t.net/ark:13030/c7sn0141m>.

[Handle] Lannom, L., "Handle System Overview", ICSTI Forum No. 30 , April 1999, <https://eric.ed.gov/?id=ED450775>.

[Kernel] Kunze, J., "A Metadata Kernel for Electronic Permanence", Journal of Digital Information Vol 2, Issue 2, ISSN 1368-7506 , January 2002, <https://n2t.net/ark:13030/c7rr1pm49>.

[N2T] ARK Alliance, "Name-to-Thing Resolver", August 2006, <https://n2t.net>.

[NAANregistry]
ARKs.org, "NAAN Registry", 2019, <https://cdluc3.github.io/naan_reg_priv/>.

[NAANrequest]
ARKs.org, "NAAN Request Form", 2018, <https://n2t.net/e/naan_request>.

[NLMPerm] Byrnes, M., "Permanence Levels and the Archives for NLM's Permanent Web Documents", March 2005, <https://www.nlm.nih.gov/pubs/techbull/ma05/ma05_archive.html>.

[NOID] Kunze, J., "Nice Opaque Identifiers", April 2006, <https://arks.org/resources/noid/>.

[PStatements] Kunze, J., "Persistence statements: describing digital stickiness", October 2016, <https://n2t.net/ark:13030/c7833mx7t>.

[PURL] Shafer, K., "Introduction to Persistent Uniform Resource Locators", 1996, <https://www.internetsociety.org/inet96/proceedings/a4/a4_1.htm>.

[RFC0854] Postel, J. and J. Reynolds, "Telnet Protocol Specification", STD 8, RFC 854, DOI 10.17487/RFC0854, May 1983, <https://www.rfc-editor.org/rfc/rfc854>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/rfc/rfc1034>.

[RFC2141] Moats, R., "URN Syntax", RFC 2141, DOI 10.17487/RFC2141, May 1997, <https://www.rfc-editor.org/rfc/rfc2141>.

[RFC2288] Lynch, C., Preston, C., and R. Daniel, "Using Existing Bibliographic Identifiers as Uniform Resource Names", RFC 2288, DOI 10.17487/RFC2288, February 1998, <https://www.rfc-editor.org/rfc/rfc2288>.

[RFC2611] Daigle, L., van Gulik, D., Iannella, R., and P. Faltstrom, "URN Namespace Definition Mechanisms", BCP 33, RFC 2611, DOI 10.17487/RFC2611, June 1999, <https://www.rfc-editor.org/rfc/rfc2611>.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/rfc/rfc2616>.

[RFC2822] Resnick, P., Ed., "Internet Message Format", RFC 2822, DOI 10.17487/RFC2822, April 2001, <https://www.rfc-editor.org/rfc/rfc2822>.

[RFC2915] Mealling, M. and R. Daniel, "The Naming Authority Pointer (NAPTR) DNS Resource Record", RFC 2915, DOI 10.17487/RFC2915, September 2000, <https://www.rfc-editor.org/rfc/rfc2915>.

[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/rfc/rfc3986>.

[RFC5013] Kunze, J. and T. Baker, "The Dublin Core Metadata Element Set", RFC 5013, DOI 10.17487/RFC5013, August 2007, <https://www.rfc-editor.org/rfc/rfc5013>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <https://www.rfc-editor.org/rfc/rfc6454>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/rfc/rfc8615>.

[shoulderrequest] ARKs.org, "Shoulder Request Form", 2021, <https://docs.google.com/forms/d/10J2VxsaeQG-IpkqZ6wpqAKqt8hYnMSf4bxdL8ktI-to>.

[SPT] Kunze, J., "What is Suffix Passthrough?", May 2021, <https://arks.org/about/ark-suffix-passthrough/>.

[THUMP] Gamiel, K. and J. Kunze, "The HTTP URL Mapping Protocol", August 2007, <https://www.ietf.org/archive/id/draft-kunze-thump-03.txt>.

Appendix A. ARK Maintenance Agency: arks.org

The ARK Maintenance Agency [ARKagency] at arks.org has several functions.

* To manage the registry of organizations that will be assigning ARKs. Organizations can request or update a NAAN by filling out the NAAN Request Form [NAANrequest].

* To be a clearinghouse for information about ARKs, such as best practices, introductory documentation, tutorials, community forums, etc. These supplemental resources help ARK implementors in high-level applications across different sectors and disciplines, and with a variety of metadata standards.

* To be a locus of discussion about future versions of the ARK specification.

Appendix B. Looking up NMAs Distributed via DNS

This subsection introduces an older method for looking up NMAs that is based on the method for discovering URN resolvers described in [RFC2915]. It relies on querying the DNS system for Name Authority Pointer (NAPTR) records that mirror the contents of the plain text [NAANregistry] database. A query is submitted to DNS asking for a list of resolvers that match a given NAAN. DNS distributes the query to the particular DNS servers that can best provide the answer, unless the answer can be found more quickly in a local DNS cache as a side-effect of a recent query. Responses come back inside NAPTR records. The normal result is one or more candidate NMAs.

In its full generality the [RFC2915] algorithm ambitiously accommodates a complex set of preferences, orderings, protocols, mapping services, regular expression rewriting rules, and DNS record types. This subsection proposes a drastic simplification of it for the special case of ARK mapping authority discovery. The simplified algorithm is called Maptr. It uses only one DNS record type (NAPTR) and restricts most of its field values to constants. The following hypothetical excerpt from a DNS data file for the NAAN known as 12026 shows three example NAPTR records ready to use with the Maptr algorithm.

```
12026.ark.arpa.
;; US Library of Congress
;;       order pref flags service regexp  replacement
 IN NAPTR  0     0   "h"  "ark"   "USLC"  lhc.nlm.nih.gov:8080
 IN NAPTR  0     0   "h"  "ark"   "USLC"  foobar.zaf.org
 IN NAPTR  0     0   "h"  "ark"   "USLC"  sneezy.dopey.com
```

All the fields are held constant for Maptr except for the "flags", "regexp", and "replacement" fields. The "service" field contains the constant value "ark" so that NAPTR records participating in the Maptr algorithm will not be confused with other NAPTR records. The "order" and "pref" fields are held to 0 (zero) and otherwise ignored for now; the algorithm may evolve to use these fields for ranking decisions when usage patterns and local administrative needs are better understood.

When a Maptr query returns a record with a flags field of "h" (for host, a Maptr extension to the NAPTR flags), the replacement field contains the NMA (host) of an ARK service provider. When a query returns a record with a flags field of "" (the empty string), the client needs to submit a new query containing the domain name found

in the replacement field. This second sort of record exploits the distributed nature of DNS by redirecting the query to another domain name. It looks like this.

```
12345.ark.arpa.
;; Digital Library Consortium
;;       order pref flags service regexp replacement
 IN NAPTR  0     0    ""  "ark"     ""   dlc.spct.org.
```

Here is the Maptr algorithm for ARK mapping authority discovery. In it replace <NAAN> with the NAAN from the ARK for which an NMA is sought.

1. Initialize the DNS query: type=NAPTR, query=<NAAN>.ark.arpa.

2. Submit the query to DNS and retrieve (NAPTR) records, discarding any record that does not have "ark" for the service field.

3. All remaining records with a flags fields of "h" contain candidate NMAs in their replacement fields. Set them aside, if any.

4. Any record with an empty flags field ("") has a replacement field containing a new domain name to which a subsequent query should be redirected. For each such record, set query=<replacement> then go to step (2). When all such records have been recursively exhausted, go to step (5).

5. All redirected queries have been resolved and a set of candidate NMAs has been accumulated from steps (3). If there are zero NMAs, exit -- no mapping authority was found. If there is one or more NMA, choose one using any criteria you wish, then exit.

A Perl script that implements this algorithm is included here.

```
#!/usr/bin/env perl

use Net::DNS;                           # include simple DNS package
my $qtype = "NAPTR";                    # initialize query type
my $naa = shift;                        # get NAAN script argument
my $mad = new Net::DNS::Resolver;       # mapping authority discovery

&maptr("$naa.ark.arpa");                # call maptr - that's it

sub maptr {                             # recursive maptr algorithm
        my $dname = shift;              # domain name as argument
        my ($rr, $order, $pref, $flags, $service, $regexp,
                $replacement);
        my $query = $mad->query($dname, $qtype);
        return                          # non-productive query
                if (! $query || ! $query->answer);
        foreach $rr ($query->answer) {
                next                    # skip records of wrong type
                        if ($rr->type ne $qtype);
                ($order, $pref, $flags, $service, $regexp,
                        $replacement) = split(/\s/, $rr->rdatastr);
                if ($flags eq "") {
                        &maptr($replacement);   # recurse
                } elsif ($flags eq "h") {
                        print "$replacement\n"; # candidate NMA
                }
        }
}
```

Authors' Addresses

John A. Kunze
Drexel University
United States of America
Email: jakkbl@gmail.com

Emmanuelle Bermティs
テ営ole nationale des Chartes
65 Rue de Richelieu
75002 Paris
France
Email: emmanuelle.bermes@chartes.psl.eu