The ARK Persistent Identifier Scheme

(http://www.ietf.org/internet-drafts/draft-kunze-ark-14.txt)

Status of this Document

Abstract

   The ARK (Archival Resource Key) naming scheme is designed to
   facilitate the high-quality and persistent identification of
   information objects. A founding principle of the ARK is that
   persistence is purely a matter of service and is neither inherent in
   an object nor conferred on it by a particular naming syntax. The best
   that an identifier can do is to lead users to the services that
   support persistence. The term ARK itself refers both to the scheme
   and to any single identifier that conforms to it. An ARK has five
   components:

J. Kunze

```
[http://NMAH/]ark:/NAAN/Name[Qualifier]
```

an optional and mutable Name Mapping Authority Hostport, the "ark:"
label, the Name Assigning Authority Number (NAAN), the assigned Name,
and an optional and possibly mutable Qualifier supported by the NMA.
The NAAN and Name together form the immutable persistent identifier
for the object. An ARK is a special kind of URL that connects users
to three things: the named object, its metadata, and the provider's
promise about its persistence. When entered into the location field
of a Web browser, the ARK leads the user to the named object. That
same ARK, followed by a single question mark ('?'), returns a brief
metadata record that is both human- and machine-readable. When the
ARK is followed by dual question marks ('??'), the returned metadata
contains a commitment statement from the current provider. Tools
exist for minting, binding, and resolving ARKs.

[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 [URNSYN] 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 has the unique advantage of working with
   unmodified web browsers. ARKs represent an approach that attempts to
   build on the strengths and to avoid the weaknesses of the other
   schemes.

   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]
   being a particularly dazzling and elegant example. What is often
   forgotten is that maintenance of an indirection table is the
   overwhelming and unavoidable cost to the organization providing
   persistence, and the 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 is an
   indication of how unsuited most organizations are to the task of
   table maintenance and to the overall challenge of digital permanence.

   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 marketplace 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 be actual liabilities for
persistence because they create short- and long-term dependencies for
every object access on complex, special-purpose local and global
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 the
entire internet community needs (the DNS, web servers, and web
browsers) and that one can reasonably expect many others to help
carry forward into the technologically evolving future.

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 is just
a URL, distinguished by its form, that provides some of the necessary
conditions for credible persistence. An ARK invites access to not
one, but to three things: to the object, to its metadata, and to a
nuanced statement of commitment from the provider regarding the
object. Existence of the two extra services can be probed
automatically by appending either `?' or `??' 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 for the large majority of a long-term
object's archival lifetime. The 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 of 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 several simple algorithms (based on the URN model) are
defined 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 wisely.

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. 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. Note that over time, current providers have
nothing to do with the intentions of the original assigners of names.

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 that reflect brief life cycle periods such as the
address of a short stay in a filesystem hierarchy. 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 access is the
central duty of an ARK. 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 catastrophic loss of the object, 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 summed up as the "HTTP
404 Not Found" problem - would not have been addressed.

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 do. There
is little that we can do about someone else's stuff except encourage
them to find or become providers 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 services 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
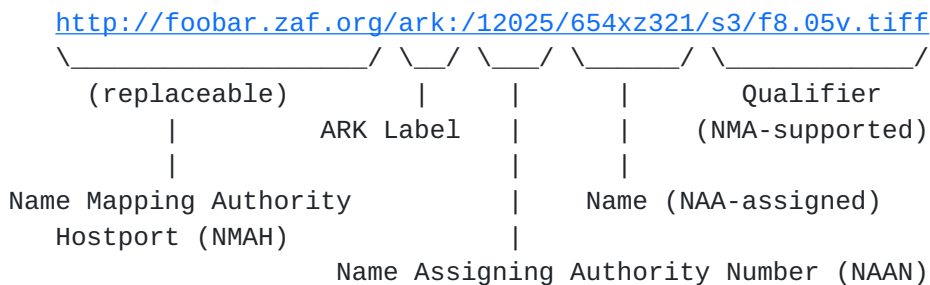   (e.g., a cataloging or other metadata record) that vouches for it.

   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 [URI]. 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.

```
    http://foobar.zaf.org/ark:/12025/654xz321/s3/f8.05v.tiff
    _____/ \__/ \___/ _____/ _____/
       (replaceable)        |     |      |        Qualifier
             |         ARK Label   |      |     (NMA-supported)
             |                     |      |
      Name Mapping Authority       |    Name (NAA-assigned)
         Hostport (NMAH)           |
                          Name Assigning Authority Number (NAAN)
```

The ARK syntax can be summarized,

                [http://NMAH/]ark:/NAAN/Name[Qualifier]

where the NMAH and Qualifier parts are in brackets to indicate that they are optional.

[2.1](#). The Name Mapping Authority Hostport (NMAH)

Before the "ark:" label may appear an optional Name Mapping Authority Hostport (NMAH) that is a temporary address where ARK service requests may be sent. It consists of "http://" (or any service specification valid for a URL) followed by an Internet hostname or hostport combination having the same format and semantics as the hostport part of a URL. The most important thing about the NMAH is that it is "identity inert" from the point of view of object identification. In other words, ARKs that differ only in the optional NMAH part identify the same object. Thus, for example, the following three ARKs are synonyms for just one information object:

                http://loc.gov/ark:/12025/654xz321
             http://rutgers.edu/ark:/12025/654xz321
                          ark:/12025/654xz321

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. In an ideal world, 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 NMAH part makes an ARK into an actionable URL. As with many
internet parameters, it is helpful to approach the NMAH being liberal
in what you accept and conservative in what you propose. From the
recipient's point of view, the NMAH 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 NMAH 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 NMAH
should 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 NMAH
eventually to stop working and require replacement with the NMAH 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
[DNS] 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 "http://" and an NMAH to an ARK is a way
of creating an actionable identifier by a method that is itself
temporary. Assuming that infrastructure supporting [HTTP]
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. In
a URL found in the wild, the string, "ark:/", 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.

Only a current provider can say for certain what sort of commitment
it intends, and the ARK label suggests that you can query the NMAH
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,

                 [http://NMAH/]ark:/NAAN/Name[Qualifier]

the part of the ARK directly following the "ark:" is the Name
Assigning Authority Number (NAAN) enclosed in `/' (slash) characters.
This part is always required, as it identifies the organization that
originally assigned the Name of the object. It is used to discover a
currently valid NMAH and to provide top-level partitioning of the
space of all ARKs. NAANs are registered in a manner similar to URN
Namespaces, but they are pure numbers consisting of 5 digits or 9
digits. Thus, the first 100,000 registered NAAs fit compactly into
the 5 digits, and if growth warrants, the next billion fit into the 9
digit form. In either case the fixed odd numbers of digits helps
reduce the chances of finding a NAAN out of context and confusing it
with nearby quantities such as 4-digit dates.

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 of an ARK
namespace.

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 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.
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.

2.5. 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 ARK 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,

        http://foobar.zaf.org/ark:/12025/654xz321/s3/f8.05v.tiff

the string "/s3/f8" is a ComponentPath and the string ".05v.tiff" 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:/12025/654/xz/321

is equivalent to publishing three ARKs,

                    ark:/12025/654/xz/321
                    ark:/12025/654/xz
                    ark:/12025/654

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:/12025/654_xz_321
                    ark:/12025/654_xz
                    ark:/12025/654xz321
                    ark:/12025/654xz
                    ark:/12025/654

Note that slashes around the ARK's NAAN (/12025/ 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; 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 period after Name, the piece to its
left is a base 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:/12025/654.24
                        ark:/12025/xz4/654.24
                        ark:/12025/654.20v.78g.f55

   There are two main rules. First, if two ARKs share the same base
   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:/12025/654.20v.78g.f55
                        ark:/12025/654.321xz
                        ark:/12025/654.44

   Second, publishing an ARK with a suffix implies the existence of at
   least one variant identified by the ARK without its suffix. The ARK
   otherwise permits no further assumptions about what variants might
   exist. So publishing the ARK,

                        ark:/12025/654.20v.78g.f55

   is equivalent to publishing the four ARKs,

                        ark:/12025/654.20v.78g.f55
                        ark:/12025/654.20v.78g
                        ark:/12025/654.20v
                        ark:/12025/654

   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. Multiple suffixes should be arranged in sorted order
   (pure ASCII collating sequence) at the end of an ARK.

2.6. Character Repertoires

   The Name and Qualifier parts are strings of visible ASCII characters
   and should be less than 128 bytes in length. The length restriction
   keeps the ARK short enough to append ordinary ARK request strings
   without running into transport restrictions (e.g., within HTTP GET
   requests). Characters may be letters, digits, or any of these six

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 NMAH, hyphens are "identity inert" in comparing ARKs for
equivalence. For example, the following ARKs are equivalent for
purposes of comparison and ARK service access:

<div align="center">

ark:/12025/65-4-xz-321

http://sneezy.dopey.com/ark:/12025/654--xz32-1

ark:/12025/654xz321

</div>

The `%' character is reserved for %-encoding all other octets that
would appear in the ARK string, in the same manner as for URIs [URI].
A %-encoded octet consists of a `%' followed by two hex digits; for
example, "%7d" stands in for `}'. Lower case hex digits are
preferred to reduce the chances of false acronym recognition; thus it
is better to 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 [URNBIB].

2.7. 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
NMAHs), 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 NMAH), the
NMAH never participates in such comparisons.

Normalization of an ARK for the purpose of octet-by-octet equality
comparison with another ARK consists of four steps. First, any upper

case letters in the "ark:" label and the two characters following a
`%' are converted to lower case. The case of all other letters in

the ARK string must be preserved. Second, any NMAH part is removed
(everything from an initial "http://" up to the next slash) and all
hyphens are removed.

Third, 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. Finally, if there are any
components with a period on the left and a slash on the right, either
the component and the preceding period must be moved to the end of
the Name part or the ARK must be thrown out as malformed.

The fourth and final step is to arrange the suffixes in ASCII
collating sequence (that is, to sort them) and to remove duplicate
suffixes, if any. It is also permissible to throw out ARKs for which
the suffixes are not sorted.

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

3.1. ARKS Embedded in Language

The ARK has different goals from the URI, so it has different
character set requirements. Because linguistic constructs imperil
persistence, for ARKs non-ASCII character support is unimportant.
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. Add the goal of
making an undelimited ARK recognizable in running prose, as in
ark:/12025/=@_22*$, and certain punctuation characters (e.g., comma,

period) end up being excluded from the ARK lest the end of a phrase
or sentence be mistaken for part of the ARK.

This consideration has more direct effect on ARK usability in a
natural language context than it has on ARK persistence. The same is
true of the rule preventing hyphens from having lexical significance.
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
as insignificant reduces the possibility of users transcribing
identifiers that will have been broken through unpredictable
hyphenation by word processors. Any measure that reduces user
irritation with an identifier will increase its chances of survival.

3.2. 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.

3.3. 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.

3.4. 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 NMAHs.

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 consortium [N2T] of like-minded
organizations with the purpose of providing mutual preservation
support. The first goal of such a consortium 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 consortium 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://" 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 NMAHs
(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.

3.5. 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:/12025/" are under control of the NAA registered under 12025,
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 [URNNID].

To register for a NAAN, please read about the mapping authority
discovery file in the next section and send email to ark@cdlib.org.

3.6.   NAAN Namespace Management

   Every NAA must have a namespace management strategy. A time-honored
   technique is to hierarchically partition a namespace into
   subnamespaces using prefixes that guarantee non-collision of names in
   different partition. This practice is strongly encouraged for all
   NAAs, especially when subnamespace management will be delegated to
   other departments, units, or projects within an organization. For
   example, with a NAAN that is assigned to a university and managed by
   its main library, care should be taken to reserve semantically opaque
   prefixes that will set aside large parts of the unused namespace for
   future assignments. Prefix-based partition management is an
   important responsibility of the NAA.

   This sort of delegation by prefix is well-used in the formation of
   DNS names and ISBN identifiers. An important difference is that 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, borrowed from
   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.

   Reasonable prefixes for assigned names usually consist of consonants
   and digits and are 1-5 characters in length. For example, the
   constant prefix "x9t" might be delegated to a book digitization
   project that creates identifiers such as

             http://444.berkeley.edu/ark:/28722/x9t38rk45c

   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 prefix
   such as "fk..." 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.

3.7. 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).

4. Finding a Name Mapping Authority

In order to derive an actionable identifier (these days, a URL) from
an ARK, a hostport (hostname or hostname plus port combination) for a
working Name Mapping Authority (NMA) must be found. An NMA is a
service that is able to respond to the three basic ARK service
requests. Relying on registration and client-side discovery, NMAs
make known which NAAs' identifiers they are willing to service.

Upon encountering an ARK, a user (or client software) looks inside it
for the optional NMAH part (the hostport of the NMA's ARK service).
If it contains an NMAH that is working, this NMAH discovery step may
be skipped; the NMAH effectively uses the beginning of an ARK to
cache the results of a prior mapping authority discovery process. If
a new NMAH needs to 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 NMAHs that
service ARKs issued by the identified NAA. The global database is
key, and two specific methods for querying it are given in this
section.

A third very promising method, called the Name-to-Thing [N2T]
Resolver, is being explored. It is a low-cost, highly stable,
consortially maintained NMAH that simply exists to support actionable
HTTP-based URLs for as long as HTTP is used. One of its big
advantages over the other two methods and the URN, Handle, DOI, and
PURL methods, is that N2T addresses the namespace splitting problem.
When objects maintained by one NMA are inherited by more than one
successor NMA, until now one of those successors would be required to
maintain forwarding tables on behalf of the other successors.

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 NMAH.

At the time of issuance, at least one NMAH 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
the National Library of Medicine 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.

4.1. Looking Up NMAHs in a Globally Accessible File

This subsection describes a way to look up NMAHs using a simple name
authority table represented as a plain text file. For efficient
access the file may be stored in a local filesystem, but it needs to
be reloaded periodically to incorporate updates. It is not expected
that the size of the file or frequency of update should impose an
undue maintenance or searching burden any time soon, for even
primitive linear search of a file with ten-thousand NAAs is a
subsecond operation on modern server machines. The proposed file
strategy is similar to the /etc/hosts file strategy that supported
Internet host address lookup for a period of years before the advent
of DNS.

The name authority table file is updated on an ongoing basis and is
available for copying over the internet from the California Digital
Library at http://www.cdlib.org/inside/diglib/ark/natab and from a

number of mirror sites. The file contains comment lines (lines that
begin with `#') explaining the format and giving the file's
modification time, reloading address, and NAA registration
instructions. There is even a Perl script that processes the file
embedded in the file's comments. As of February 2006, currently
registered Name Assigning Authorities are:

```
     12025          National Library of Medicine
     12026          Library of Congress
     12027          National Agriculture Library
     13030          California Digital Library
     13038          World Intellectual Property Organization
     20775          University of California San Diego
     29114          University of California San Francisco
     28722          University of California Berkeley
     21198          University of California Los Angeles
     15230          Rutgers University
     13960          Internet Archive
     64269          Digital Curation Centre
     62624          New York University
     67531          University of North Texas
     27927          Ithaka Electronic-Archiving Initiative
     12148          Bibliotheque nationale de France / National Library of
France
     78319          Google
     88435          Princeton University
     78428          University of Washington
     89901          Archives of Region of Vastra Gotaland and City of
Gothenburg, Sweden
     80444          Northwest Digital Archives
     25593          Emory University
     25031          University of Kansas
     17101          Centre for Ecology & Hydrology, UK
     65323          University of Calgary
```

A snapshot of the name authority table file appears in an appendix.

4.2. Looking up NMAHs Distributed via DNS

This subsection introduces a method for looking up NMAHs that is
based on the method for discovering URN resolvers described in
[NAPTR]. It relies on querying the DNS system already installed in
the background infrastructure of most networked computers. 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 Name Authority Pointer (NAPTR) records.

The normal result is one or more candidate NMAHs.

In its full generality the [NAPTR] 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
hostport, a Maptr extension to the NAPTR flags), the replacement
field contains the NMAH (hostport) 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 NMAH 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 NMAHs 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 NMAHs has been accumulated from steps (3). If there
are zero NMAHs, exit - no mapping authority was found. If there
is one or more NMAH, choose one using any criteria you wish,
then exit.

A Perl script that implements this algorithm is included here.

```perl
#!/depot/bin/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 NMAH
        }
    }
}
```

The global database thus distributed via DNS and the Maptr algorithm
can easily be seen to mirror the contents of the Name Authority Table
file described in the previous section.

[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.

[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.

[5.2](#). **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 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.

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

(b) "identifier validity" - under what conditions the identifier

will be or has been re-assigned,

(c) "content invariance" - under what conditions the content of
the object is subject to change, and

(d) "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 a permanence rating framework, conceived
independently from ARKs, is given in [NLMPerm]. Under ongoing
development and limited deployment at the US National Library of
Medicine, it identifies 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 the NMA who 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.

(i) "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

(ii) "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.

5.3. Generic Description Service

   Returns a description of the object. Descriptions are returned in
   either a structured metadata format or a human readable text format;
   sometimes one format may serve both purposes. A description must at
   a minimum answer the who, what, when, and where questions 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.

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

   The HTTP URL Mapping Protocol (THUMP) is a way of taking a key (a
   kind of 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 within keyboard-based [TELNET] sessions.

   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
   NMAH 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 three most common requests correspond to three degenerate special cases that keep the user's learning and typing burden low. 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 "?". For example, entering the string,

        ark.nlm.nih.gov/12025/psbbantu?

into the browser's location field directly precipitates a request for a metadata record describing the object identified by ark:/12025/psbbantu. 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 the following example THUMP session, 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 (4-7), and the body of the server's response (8-17). The first and last lines (1 and 17) 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 http://ark.nlm.nih.gov/ark:/12025/psbbantu? HTTP/1.1
    C:
    S: HTTP/1.1 200 OK
 5  S: Content-Type: text/plain
    S: THUMP-Status: 0.1 200 OK
    S:
    S: |set: NLM | 12025/psbbantu? | 20030731
    S:         | http://ark.nlm.nih.gov/ark:/12025/psbbantu?
10  S: here: 1 | 1 | 1
    S:
    S: erc:
    S: who:    Lederberg, Joshua
    S: what:   Studies of Human Families for Genetic Linkage
15  S: when:   1974
    S: where:  http://profiles.nlm.nih.gov/BB/A/N/T/U/_/bbantu.pdf
    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 balance of the response
consists of a record set header (lines 8-10) and a single metadata
record (12-16) that comprises the ARK description service response.
The record set header identifies (8-9) who created the set, what its
title is, when it was created, and where an automated process can
access the set; it ends in a line (10) whose respective sub-elements
indicate that here in this communication the recipient can expect to
find 1 record, starting at the record numbered 1, from a set
consisting of a total of 1 record (i.e., here is the entire set,
consisting of exactly one record).

The returned record (12-16) is in the format of an Electronic
Resource Citation [ERC], which is discussed in more detail 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 is was created, and where the expression
may be found. This quartet of elements comes up again and again in
ERCs.

The third degenerate special case of an ARK request (and no other
cases will be described in this document) is the string "??",
corresponding to a minimal permanence policy request. It can be seen
in use appended to an ARK (on line 2) in the example session that
follows.

```
  1  C: [opens session]
     C: GET http://ark.nlm.nih.gov/ark:/12025/psbbantu?? HTTP/1.1
     C:
     S: HTTP/1.1 200 OK
  5  S: Content-Type: text/plain
     S: THUMP-Status: 0.1 200 OK
     S:
     S: |set: NLM | 12025/psbbantu?? | 20030731
     S:          | http://ark.nlm.nih.gov/ark:/12025/psbbantu??
 10  S: here: 1 | 1 | 1
     S:
     S: erc:
     S: who:    Lederberg, Joshua
     S: what:   Studies of Human Families for Genetic Linkage
 15  S: when:   1974
     S: where:  http://profiles.nlm.nih.gov/BB/A/N/T/U/_/bbantu.pdf
     S: erc-support:
     S: who:    USNLM
     S: what:   Permanent, Unchanging Content
 20  S: when:   20010421
     S: where:  http://ark.nlm.nih.gov/yy22948
     S: [closes session]
```

Again, a single metadata record (lines 12-21) is returned, but it
consists of two segments. The first segment (12-16) gives the same
basic citation information as in the previous example. It is
returned in order to establish context for the persistence
declaration in the second segment (17-21).

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.

7. Overview of Electronic Resource Citations (ERCs)

An Electronic Resource Citation (or ERC, pronounced e-r-c) [ERC] is a
kind of object description that uses Dublin Core Kernel [Kernel]
metadata elements. The ERC with Kernel elements provides a simple,
compact, and printable record for holding data associated with an
information resource. By design, Kernel metadata balances the needs
for expressive power, very simple machine processing, and direct
human manipulation.

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.

In the current Internet, it is natural seriously to consider using
XML as an exchange format because of predictions that it will obviate
many ad hoc formats and programs, and unify much of the world's
information under one reliable data structuring discipline that is
easy to generate, verify, parse, and render. It appears, however,
that XML is still only catching on after years of standards work and
implementation experience. The reasons for it are unclear, but for
now very simple XML interpretation is still out of reach. Another
important caution is that XML structures are hard on the eyeballs,
taking up an amount of display (and page) space that significantly
exceeds that of traditional formats. Until these conflicts with ERC
principle are resolved, XML is not a first choice for representing
ERCs. Borrowing instead from the data structuring format that
underlies the successful spread of email and web services, the first
ERC format uses [ANVL], which is based on email and HTTP headers
[RFC822]. 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.

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 [DCORE] 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 location, 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.

7.1. ERC Syntax

An ERC record is a sequence of metadata elements ending in a blank
line. An element consists of a label, a colon, and an optional
value. Here is an example of a record with five elements.

        erc:
        who: Gibbon, Edward
        what: The Decline and Fall of the Roman Empire
        when: 1781
        where: http://www.ccel.org/g/gibbon/decline/

A long value may be folded (continued) onto the next line by
inserting a newline and indenting the next line. A value can be thus
folded across multiple lines. Here are two example elements, each
folded across four lines.

        who/created: University of California, San Francisco, AIDS
            Program at San Francisco General Hospital | University
            of California, San Francisco, Center for AIDS Prevention
            Studies
        what/Topic:
             Heart Attack | Heart Failure
            | Heart
                            Diseases

An element value folded across several lines is treated as if the
lines were joined together on one long line. For example, the second
element from the previous example is considered equivalent to

        what/Topic: Heart Attack | Heart Failure | Heart Diseases

An element value may contain multiple values, each one separated from
the next by a `|' (pipe) character. The element from the previous
example contains three values.

For annotation purposes, any line beginning with a `#' (hash)
character is treated as if it were not present; this is a "comment"

line (a feature not available in email or HTTP headers). For
example, the following element is spread across four lines and

contains two values:

```
    what/Topic:
        Heart Attack
#     | Heart Failure  -- hold off until next review cycle
        | Heart Diseases
```

7.2. ERC Stories

An ERC record is organized into one or more distinct segments, where
where each segment tells a story about a different aspect of the
information resource. A segment boundary occurs whenever a segment
label (an element beginning with "erc") is encountered. The basic
label "erc:" introduces the story of an object's expression (e.g.,
its publication, installation, or performance). The label "erc-
about:" introduces the story of an object's content (what it is
about) and "erc-support:" introduces the story of a support
commitment made to it. A story segment that concerns the ERC itself
is introduced by the label "erc-from:". It is an important segment
that tells the story of the ERC's provenance. Elements beginning
with "erc" are reserved for segment labels and their associated story
types. From an earlier example, here is an ERC with two segments.

```
    erc:
    who:    Lederberg, Joshua
    what:   Studies of Human Families for Genetic Linkage
    when:   1974
    where:  http://profiles.nlm.nih.gov/BB/A/N/T/U/_/bbantu.pdf
    erc-support:
    who:    NIH/NLM/LHNCBC
    what:   Permanent, Unchanging Content
    # Note to ops staff:  date needs verification.
    when:   2001 04 21
    where:  http://ark.nlm.nih.gov/yy22948
```

Segment stories are told according to journalistic tradition. While
any number of pertinent elements may appear in a segment, priority is
placed on answering the questions who, what, when, and where at the
beginning of each segment so that readers can make the most important
selection or rejection decisions as soon as possible. To make things
simple, the listed ordering of the questions is maintained in each
segment (as it happens most people who have been exposed to this
story telling technique are already familiar with the above
ordering).

The four questions are answered by using corresponding element
labels. The four element labels can be re-used in each story

segment, but their meaning changes depending on the segment (the
story type) in which they appear. In the example above, "who" is
first used to name a document's author and subsequently used to name

the permanence guarantor (provider). Similarly, "when" first lists
the date of object creation and in the next segment lists the date of
a commitment decision. Four labels appearing across three segments
effectively map to twelve semantically distinct elements. Distinct
element meanings are mapped to Dublin Core elements in a later
section.

7.3. The ERC Anchoring Story

Each ERC contains an anchoring story. It is usually the first
segment labeled "erc:" and it concerns an "anchoring" expression of
the object. An "anchoring" expression is the one that a provider
deemed the most suitable basic referent given the audience and
application for which it produced the ERC. If it sounds like the
provider has great latitude in choosing its anchoring expression, it
is because it does. A typical anchoring story in an ERC for a born-
digital document would be the story of the document's release on a
web site; such a document would then be the anchoring expression.

An anchoring story need not be the central descriptive goal of an ERC
record. For example, a museum provider may create an ERC for a
digitized photograph of a painting but choose to anchor it in the
story of the original painting instead of the story of the electronic
likeness; although the ERC may through other segments prove to be
centrally concerned with describing the electronic likeness, the
provider may have chosen this particular anchoring story in order to
make the ERC visible in a way that is most natural to patrons (who
would find the Mona Lisa under da Vinci sooner than they would find
it under the name of the person who snapped the photograph or scanned
the image). In another example, a provider that creates an ERC for a
dramatic play as an abstract work has the task of describing a piece
of intangible intellectual property. To anchor this abstract object
in the concrete world, if only through a derivative expression, it
makes sense for the provider to choose a suitable printed edition of
the play as the anchoring object expression (to describe in the
anchoring story) of the ERC.

The anchoring story has special rules designed to keep ERC processing
simple and predictable. Each of the four basic elements (who, what,
when, and where) must be present, unless a best effort to supply it
fails. In the event of failure, the element still appears but a
special value (described later) is used to explain the missing value.
While the requirement that each of the four elements be present only
applies to the anchoring story segment, as usual these elements
appear at the beginning of the segment and may only be used in the
prescribed order. A minimal ERC would normally consist of just an
anchoring story and the element quartet, as illustrated in the next
example.

```
        erc:
        who:   National Research Council
        what:  The Digital Dilemma
        when:  2000
        where: http://books.nap.edu/html/digital%5Fdilemma
```

A minimal ERC can be abbreviated so that it resembles a traditional
compact bibliographic citation that is nonetheless completely machine
processable. The required elements and ordering makes it possible to
eliminate the element labels, as shown here.

```
        erc: National Research Council | The Digital Dilemma | 2000
                | http://books.nap.edu/html/digital%5Fdilemma
```

7.4. ERC Elements

As mentioned, the four basic ERC elements (who, what, when, and
where) take on different specific meanings depending on the story
segment in which they are used. By appearing in each segment, albeit
in different guises, the four elements serve as a valuable mnemonic
device - a kind of checklist - for constructing minimal story
segments from scratch. Again, it is only in the anchoring segment
that all four elements are mandatory.

Here are some mappings between ERC elements and Dublin Core [DCORE]
elements.

```
        Segment      ERC Element      Equivalent Dublin Core Element
      ---------     -----------      ------------------------------
         erc           who          Creator/Contributor/Publisher
         erc           what                 Title
         erc           when                 Date
         erc           where                Identifier
      erc-about        who                  <none>
      erc-about        what                 Subject
      erc-about        when                 Coverage (temporal)
      erc-about        where                Coverage (spatial)
```

The basic element labels may also be qualified to add nuances to the
semantic categories that they identify. Elements are qualified by
appending a `/' (slash) and a qualifier term. Often qualifier terms
appear as the past tense form of a verb because it makes re-using
qualifiers among elements easier.

```
        who/published:  ...
        when/published: ...
        where/published: ...
```

Using past tense verbs for qualifiers also reminds providers and
   recipients that element values contain transient assertions that may

have been true once, but that tend to become less true over time.
Recipients that don't understand the meaning of a qualifier can fall
back onto the semantic category (bucket) designated by the
unqualified element label. Inevitably recipients (people and
software) will have diverse abilities in understanding elements and
qualifiers.

Any number of other elements and qualifiers may be used in
conjunction with the quartet of basic segment questions. The only
semantic requirement is that they pertain to the segment's story.
Also, it is only the four basic elements that change meaning
depending on their segment context. All other elements have meaning
independent of the segment in which they appear. If an element label
stripped of its qualifier is still not recognized by the recipient, a
second fall back position is to ignore it and rely on the four basic
elements.

Elements may be either Canonical, Provisional, or Local. Canonical
elements are officially recognized via a registry as part of the
metadata vernacular. All elements, qualifiers, and segment labels
used in this document up until now belong to that vernacular.
Provisional elements are also officially recognized via the registry,
but have only been proposed for inclusion in the vernacular. To be
promoted to the vernacular, a provisional element passes through a
vetting process during which its documentation must be in order and
its community acceptance demonstrated. Local elements are any
elements not officially recognized in the registry. The registry
[Kernel] is a work in progress.

Local elements can be immediately distinguishable from Canonical or
Provisional elements because all terms that begin with an upper case
letter are reserved for spontaneous local use. No term beginning
with an upper case letter will ever be assigned Canonical or
Provisional status, so it should be safe to use such terms for local
purposes. Any recipient of external ERCs containing such terms will
understand them to be part of the originating provider's local
metadata dialect. Here's an example ERC with three segments, one
local element, and two local qualifiers. The segment boundaries have
been emphasized by comment lines (which, as before, are ignored by
processors).

```
        erc:
        who: Bullock, TH | Achimowicz, JZ | Duckrow, RB
                | Spencer, SS | Iragui-Madoz, VJ
        what: Bicoherence of intracranial EEG in sleep,
                wakefulness and seizures
        when: 1997 12 00
        where: http://cogprints.soton.ac.uk/%{
                documents/disk0/00/00/01/22/index.html %}
        in: EEG Clin Neurophysiol | 1997 12 00 | v103, i6, p661-678
        IDcode: cog00000122
        # ---- new segment ----
        erc-about:
        what/Subcategory: Bispectrum | Nonlinearity | Epilepsy
                | Cooperativity | Subdural | Hippocampus | Higher moment
        # ---- new segment ----
        erc-from:
        who: NIH/NLM/NCBI
        what: pm9546494
        when/Reviewed: 1998 04 18 021600
        where: http://ark.nlm.nih.gov/12025/pm9546494?
```

The local element "IDcode" immediately precedes the "erc-about"
segment, which itself contains an element with the local qualifier
"Subcategory". The second to last element also carries the local
qualifier "Reviewed". Finally, what might be a provisional element
"in" appears near the end of the first segment. It might have been
proposed as a way to complete a citation for an object originally
appearing inside another object (such as an article appearing in a
journal or an encyclopedia).

7.5. ERC Element Values

ERC element values tend to be straightforward strings. If the
provider intends something special for an element, it will so
indicate with markers at the beginning of its value string. The
markers are designed to be uncommon enough that they would not likely
occur in normal data except by deliberate intent. Markers can only
occur near the beginning of a string, and once any octet of non-
marker data has been encountered, no further marker processing is
done for the element value. In the absence of markers the string is
considered pure data; this has been the case with all the examples
seen thus far. The fullest form of an element value with all three
optional markers in place looks like this.

```
        VALUE =    [markup_flags]    (:ccode)    ,    DATA
```

In processing, the first non-whitespace character of an ERC element
value is examined. An initial `[' is reserved to introduce a

bracketed set of markup flags (not described in this document) that
ends with `]'. If ERC data is machine-generated, each value string
may be preceded by "[]" to prevent any of its data from being

mistaken for markup flags. Once past the optional markup, the
remaining value may optionally begin with a controlled code. A
controlled code always has the form "(:ccode)", for example,

        who: (:unkn) Anonymous
        what: (:791) Bee Stings

Any string after such a code is taken to be an uncontrolled (e.g.,
natural language) equivalent. The code "unkn" indicates a
conventional explanation for a missing value (stating that the value
is unknown). The remainder of the string makes an equivalent
statement in a form that the provider deemed most suitable to its
(probably human) audience. The code "791" could be a fixed numeric
topic identifier within an unspecified topic vocabulary. Any code
may be ignored by those that do not understand it.

There are several codes to explain different ways in which a required
element's value may go missing.

        (:unac)   temporarily inaccessible
        (:unal)   unallowed, suppressed intentionally
        (:unap)   not applicable, makes no sense
        (:unas)   value unassigned (e.g., Untitled)
        (:unav)   value unavailable indefinitely
        (:unkn)   unknown (e.g., Anonymous, Inconnue)
        (:etal)   too numerous to list (I<et alia>).
        (:none)   never had a value, never will
        (:null)   explicitly empty
        (:tba)   to be assigned or announced later

Once past an optional controlled code, the remaining string value is
subjected to one final test. If the first next non-whitespace
character is a `,' (comma), it indicates that the string value is
"sort-friendly". This means that the value is (a) laid out with an
inverted word order useful for sorting items having comparably laid
out element values (items might be the containing ERC records) and
(b) that the value may contain other commas that indicate inversion
points should it become necessary to recover the value in natural
word order. Typically, this feature is used to express Western-style
personal names in family-name-given-name order. It can also be used
wherever natural word order might make sorting tricky, such as when
data contains titles or corporate names. Here are some example
elements.

          who: , van Gogh, Vincent
          who:,Howell, III, PhD, 1922-1987, Thurston
          who:, Acme Rocket Factory, Inc., The
          who:, Mao Tse Tung
          who:, McCartney, Paul, Sir,
          what:, Health and Human Services, United States Government
                  Department of, The,

   There are rules to use in recovering a copy of the value in natural
   word order, if desired. The above example strings have the following
   natural word order values, respectively.

          Vincent van Gogh
          Thurston Howell, III, PhD, 1922-1987
          The Acme Rocket Factory, Inc.
          Mao Tse Tung
          Sir Paul McCartney
          The United States Government Department of Health and Human Services

[7.6](#). **ERC Element Encoding and Dates**

   Some characters that need to appear in ERC element values might
   conflict with special characters used for structuring ERCs, so there
   needs to be a way to include them as literal characters that are
   protected from special interpretation. This is accomplished through
   an encoding mechanism that resembles the %-encoding familiar to [URI]
   handlers.

   The ERC encoding mechanism also uses `%', but instead of taking two
   following hexadecimal digits, it takes one non-alphanumeric character
   or two alphabetic characters that cannot be mistaken for hex digits.
   It is designed not to be confused with normal web-style %-encoding.
   In particular it can be decoded without risking unintended decoding
   of normal %-encoded data (which would introduce errors). Here are
   the one-character (non-alphanumeric) ERC encoding extensions.

          ERC   Purpose
          ---   --
          %!   decodes to the element separator `|'
          %%   decodes to a percent sign `%'
          %.   decodes to a comma `,'
          %_   a non-character used as syntax shim
          %{   a non-character that begins an expansion block
          %}   a non-character that ends an expansion block

   One particularly useful construct in ERC element values is the pair
   of special encoding markers ("%{" and "%}") that indicates a
   "expansion" block. Whatever string of characters they enclose will

be treated as if none of the contained whitespace (SPACEs, TABs,
Newlines) were present. This comes in handy for writing long, multi-

part URLs in a readable way. For example, the value in

```
        where: http://foo.bar.org/node%{
                    ? db = foo
                    & start = 1
                    & end = 5
                    & buf = 2
                    & query = foo + bar + zaf
               %}
```

is decoded into an equivalent element, but with a correct and intact
URL:

```
  where:
    http://foo.bar.org/node?db=foo&start=1&end=5&buf=2&query=foo+bar+zaf
```

In a parting word about ERC element values, a commonly recurring
value type is a date, possibly followed by a time. ERC dates use the
[TEMPER] format, taking on one of the following forms:

```
    1999                   (four digit year)
    2000 12 29             (year, month, day)
    2000 12 29 235955    (year, month, day, hour, minute, second)
```

In dates, all internal whitespace is squeezed out to achieve a
normalized form suitable for lexical comparison and sorting. This
means that the following dates

```
    2000 12 29 235955              (recommended for readability)
    2000 12 29 23 59 55
    20001229 23 59 55
    20001229235955                 (normalized date and time)
```

are all equivalent. The first form is recommended for readability.
The last form (shortest and easiest to compute with) is the
normalized form. Hyphens and commas are reserved to create date
ranges and lists, for example,

```
    1996-2000                      (a range of four years)
    1952, 1957, 1969               (a list of three years)
    1952, 1958-1967, 1985          (a mixed list of dates and ranges)
    20001229-20001231              (a range of three days)
```

7.7. ERC Stub Records and Internal Support

The ERC design introduces the concept of a "stub" record, which is an
incomplete ERC record intended to be supplemented with additional

elements before being released as a standalone ERC record. A stub
ERC record has no minimum required elements. It is just a group of

elements that does not begin with "erc:" but otherwise conforms to
the ERC record syntax.

ERC stubs may be useful in supporting internal procedures using the
ERC syntax. Often they rely on the convenience and accuracy of
automatically supplied elements, even the basic ones. To be ready
for external use, however, an ERC stub must be transformed into a
complete ERC record having the usual required elements. An ERC stub
record can be convenient for metadata embedded in a document, where
elements such as location, modification date, and size - which one
would not omit from an externalized record - are omitted simply
because they are much better supplied by a computation. A separate
local administrative procedure, not defined for ERC's in general,
would effect the promotion of stubs into complete records.

While the ERC is a general-purpose container for exchange of resource
descriptions, it does not dictate how records must be internally
stored, laid out, or assembled by data providers or recipients.
Arbitrary internal descriptive frameworks can support ERCs simply by
mapping (e.g., on demand) local records to the ERC container format
and making them available for export. Therefore, to support ERCs
there is no need for a data provider to convert internal data to be
stored in an ERC format. On the other hand, any provider (such as
one just getting started in the business of resource description) may
choose to store and manipulate local data natively in the ERC format.

8. Advice to Web Clients

This section offers some advice to web client software developers.
It is hard to write about because it tries to anticipate a series of
events that might lead to native web browser support for ARKs.

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) currently stand in, and URNs, DOIs, and
PURLs have been proposed as alternatives.

The strings representing ARKs are also envisaged to appear in some of
the places where URLs currently appear: in hypertext links (where
they are not normally shown to users) and in rendered text (displayed
or printed). Internet search engines, for example, tend to include
both actionable and manifest links when listing each item found. A
normal HTML link for which the URL is not displayed looks like this.

        <a href = "http://foo.bar.org/index.htm"> Click Here <a>

The same link with an ARK instead of a URL:

```
<a href = "ark:/14697/b12345x"> Click Here <a>
```

Web browsers would in general require a small modification to
recognize and convert this ARK, via mapping authority discovery, to
the URL form.

        <a href = "http://a.b.org/ark:/14697/b12345x"> Click Here <a>

A browser that knows how to make that conversion could also
automatically detect and replace a non-working NMAH.

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.

9. 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 hostport (NMAH) 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, Z39.50).
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.

10. Authors' Addresses

John A. Kunze
California Digital Library
University of California, Office of the President
415 20th St, 4th Floor
Oakland, CA 94612-3550, USA

Fax: +1 510-893-5212
EMail: jak@ucop.edu

R. P. C. Rodgers
US National Library of Medicine
8600 Rockville Pike, Bldg. 38A
Bethesda, MD 20894, USA

   Fax: +1 301-496-0673

EMail: rodgers@nlm.nih.gov

11. References

[ANVL]   J. Kunze, B. Kahle, et al, "A Name-Value Language", work
            in progress,
            http://www.cdlib.org/inside/diglib/ark/anvlspec.pdf

[ARK]   J. Kunze, "Towards Electronic Persistence Using ARK
            Identifiers", Proceedings of the 3rd ECDL Workshop on Web
            Archives, August 2003, (PDF)
            http://bibnum.bnf.fr/ecdl/2003/proceedings.php?f=kunze

[DCORE]   Dublin Core Metadata Initiative, "Dublin Core Metadata
            Element Set, Version 1.1: Reference Description", July
            1999, http://dublincore.org/documents/dces/.

[DNS]   P.V. Mockapetris, "Domain Names - Concepts and
            Facilities", RFC 1034, November 1987.

[DOI]   International DOI Foundation, "The Digital Object
            Identifier (DOI) System", February 2001,
            http://dx.doi.org/10.1000/203.

[ERC]   J. Kunze, "A Metadata Kernel for Electronic Permanence",
            Journal of Digital Information, Vol 2, Issue 2, January
            2002, ISSN 1368-7506, (PDF)
            http://jodi.ecs.soton.ac.uk/Articles/v02/i02/Kunze/

[Handle]   L. Lannom, "Handle System Overview", ICSTI Forum, No. 30,
            April 1999, http://www.icsti.org/forum/30/#lannom

[HTTP]   R. Fielding, et al, "Hypertext Transfer Protocol --
            HTTP/1.1", RFC 2616, June 1999.

[Kernel]   Dublin Core Metadata Initiative, "Kernel Metadata Working
            Group", http://dublincore.org/groups/kernel/

[MD5]   R. Rivest, "The MD5 Message-Digest Algorithm", RFC 1321,
            April 1992.

[N2T]   CDL, "Name-to-Thing Resolover", work in progress, August
            2006, http://n2t.info

[NAPTR]   M. Mealling, Daniel, R., "The Naming Authority Pointer
            (NAPTR) DNS Resource Record", RFC 2915, September 2000.

[NLMPerm]   M. Byrnes, "Defining NLM's Commitment to the Permanence of
            Electronic Information", ARL 212:8-9, October 2000,

http://www.arl.org/newsltr/212/nlm.html

   [NOID]   J. Kunze, "Nice Opaque Identifiers", February 2005,
               http://www.cdlib.org/inside/diglib/ark/noid.pdf

   [PURL]   K. Shafer, et al, "Introduction to Persistent Uniform
               Resource Locators", 1996,
               http://purl.oclc.org/OCLC/PURL/INET96

   [RFC822]   D. Crocker, "Standard for the format of ARPA Internet text
               messages", RFC 822, August 1982.

   [TELNET]   J. Postel, J.K. Reynolds, "Telnet Protocol Specification",
               RFC 854, May 1983.

   [TEMPER]   J. Kunze, "Temporal Enumerated Ranges", work in progress,
               http://www.cdlib.org/inside/diglib/ark/temperspec.pdf

   [THUMP]   K. Gamiel, J. Kunze, "The HTTP URL Mapping Protocol", work
               in progress, http://www.ietf.org/internet-drafts/draft-
               kunze-thump-00.txt

   [URI]   T. Berners-Lee, et al, "Uniform Resource Identifiers
               (URI): Generic Syntax", RFC 2396, August 1998.

   [URNBIB]   C. Lynch, et al, "Using Existing Bibliographic Identifiers
               as Uniform Resource Names", RFC 2288, February 1998.

   [URNSYN]   R. Moats, "URN Syntax", RFC 2141, May 1997.

   [URNNID]   L. Daigle, et al, "URN Namespace Definition Mechanisms",
               RFC 2611, June 1999.

12. Appendix: ARK Implementations

   Currently, the primary implementation activity is at the California
   Digital Library (CDL),

          http://ark.cdlib.org/

   housed at the University of California Office of the President, where
   over 200,000 ARKs have been assigned to objects that the CDL owns or
   controls. Some experimentation in ARKs is taking place at JSTOR, the
   Digital Curation Centre, WIPO and at the University of California's
   San Diego, San Francisco, and Berkeley campuses.

   The US National Library of Medicine (NLM) also has an experimental,
   prototype ARK service under development. It is being made available
   for purposes of demonstrating various aspects of the ARK system, but
   is subject to temporary or permanent withdrawal (without notice)
   depending upon the circumstances of the small research group

responsible for making it available. It is described at:

http://ark.nlm.nih.gov/

Comments and feedback may be addressed to rodgers@nlm.nih.gov.

13. Appendix: Current ARK Name Authority Table

This appendix contains a copy of the Name Authority Table (a file) at
the time of writing. It may be loaded into a local filesystem (e.g.,
/etc/natab) for use in mapping NAAs (Name Assigning Authorities) to
NMAHs (Name Mapping Authority Hostports). It contains Perl code that
can be copied into a standalone script that processes the table (as a
file). Because this is still a proposed file, none of the values in
it are real.

```
#
# Name Assigning Authority / Name Mapping Authority Lookup Table
#    Last change:   2007.06.05
#        Reload from:    http://ark.nlm.nih.gov/etc/natab
#        Mirrored at:    http://www.cdlib.org/inside/diglib/ark/natab
#        To register:    mailto:ark@cdlib.org?Subject=naareg
#        Process with:   Perl script at end of this file (optional)
#
# Each NAA appears at the beginning of a line with the NAA Number
# first, a colon, and an ARK or URL to a statement of naming policy
# (see http://ark.cdlib.org for an example).
# All the NMA hostports that service an NAA are listed, one per
# line, indented, after the corresponding NAA line.
#
#        National Library of Medicine
12025:  http://www.nlm.nih.gov/xxx/naapolicy.html
        ark.nlm.nih.gov USNLM
        foobar.zaf.org UCSF
#
#        Library of Congress
12026:  http://www.loc.gov/xxx/naapolicy.html
        foobar.zaf.org USLC
#
#        National Agriculture Library
12027:  http://www.nal.gov/xxx/naapolicy.html
        foobar.zaf.gov:80 USNAL
#
#        California Digital Library
13030:  http://www.cdlib.org/inside/diglib/ark/
        ark.cdlib.org CDL
#
#        World Intellectual Property Organization
13038:  http://www.wipo.int/xxx/naapolicy.html
        www.wipo.int WIPO
#
#        University of California San Diego
20775:  http://library.ucsd.edu/xxx/naapolicy.html
        ucsd.edu UCSD
#
#        University of California San Francisco
29114:  http://library.ucsf.edu/xxx/naapolicy.html
        ucsf.edu UCSF
#
#        University of California Berkeley
28722:  http://library.berkeley.edu/xxx/naapolicy.html
        berkeley.edu UCB
#
#        University of California Los Angeles
```

21198: http://library.ucla.edu/xxx/naapolicy.html
                ucla.edu UCLA
        #

```
#        Rutgers University
15230:   http://rci.rutgers.edu/xxx/naapolicy.html
         rutgers.edu RU
#
#        Internet Archive
13960:   http://www.archive.org/xxx/naapolicy.html
         archive.org IA
#
#        Digital Curation Centre
64269:   http://www.dcc.ac.uk/xxx/naapolicy.html
         dcc.ac.uk DCC
#
#        New York University
62624:   http://library.nyu.edu/xxx/naapolicy.html
         nyu.edu NYU
#
#        University of North Texas
67531:   http://www.library.unt.edu/xxx/naapolicy.html
         unt.edu UNT
#
#        Ithaka Electronic-Archiving Initiative
27927:   http://www.ithaka.org/xxx/naapolicy.html
         ithaka.org ITHAKA
#
#        Bibliothque nationale de France / National Library of France
12148:   http://www.bnf.fr/xxx/naapolicy.html
         bnf.fr BNF
#
#        Princeton University
88435:   http://diglib.princeton.edu/xxx/naapolicy.html
         princeton.edu PU
#
#        University of Washington
78428:   http://u.washington.edu/xxx/naapolicy.html
         u.washington.edu UW
#
#        Archives of Region of Vstra Gtaland and City of Gothenburg, Sweden
89901:   http://www.arkivnamnden.org/xxx/naapolicy.html
         arkivnamnden.org AVGG
#
#        Northwest Digital Archives
80444:   http://nwda.wsulibs.wsu.edu/xxx/naapolicy.html
         nwda.wsulibs.wsu.edu NWDA
#
#        Emory University
25593:   http://id.library.emory.edu/xxx/naapolicy.html
         id.library.emory.edu EMORY
#
```

```
     #       University of Kansas
  25031:  http://www.lib.ku.edu/xxx/naapolicy.html
          www.lib.ku.edu UKANSAS
```

```
       #
       #       Google
       78319:  http://www.google.com/xxx/naapolicy.html
               www.google.com GOOGLE
       #
       #    Centre for Ecology & Hydrology, UK
       17101:  http://www.ceh.ac.uk/xxx/naapolicy.html
            www.ceh.ac.uk CEH
       #
       #    University of Calgary
       65323:  http://library.ucalgary.ca/xxx/naapolicy.html
            ucalgary.ca UCALGARY
       #
       #12345: reserved for examples
       #
       #--- end of data ---
       # The following Perl script takes an NAA as argument and outputs
       # the NMAs in this file listed under any matching NAA.
       #
       # my $naa = shift;
       # while (<>) {
       #       next if (! /^$naa:/);
       #       while (<>) {
       #               last if (! /^[#\s]./);
       #               print "$1\n" if (/^\s+(\S+)/);
       #       }
       # }
       #
       # Create a g/t/nroff-safe version of this table with the UNIX command,
       #
       #       expand natab | sed 's/\\/\\\e/g' > natab.roff
       #
       # end of file
```

14. Copyright Notice

Table of Contents