Internet-Draft: draft-kunze-ark-01.txt
ARK Identifier Scheme
Expires 8 September 2001

J. Kunze
University of California (UCSF)
R. P. C. Rodgers
US National Library of Medicine
8 March 2001

The ARK Persistent Identifier Scheme

(http://www.ckm.ucsf.edu/people/jak/home/ark-01.txt)
(http://www.ckm.ucsf.edu/people/jak/home/ark-01.ps)

Status of this Document

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as ‘‘work in progress.’’

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Distribution of this document is unlimited. Please send comments to jak@ckm.ucsf.edu.

Copyright (C) The Internet Society (2001). All Rights Reserved.

Abstract

The ARK (Archival Resource Key) is a scheme intended to facilitate the persistent naming and retrieval of information objects. It comprises an identifier syntax and three services. An ARK has four components:

`ark:`[NMAH]/NAAN/Name

the prefix "ark:", the (optional and mutable) Name Mapping Authority Hostport (NMAH, where "hostport" is a hostname followed optionally by a colon and port number), the Name Assigning Authority Number (NAAN), and the assigned Name. The NAAN and Name together form the immutable persistent identifier for the object.

An ARK request is an ARK to which is appended a service request beginning with a question mark. Use of an ARK request proceeds in two steps. First, the NMAH, if not specified, is discovered based on the NAAN. Two methods for discovery are proposed: one is file based, the other based on the DNS NAPTR record. Second, the ARK request is submitted to the NMAH. Three ARK services are defined, gaining access to: (1) the object (or a sensible substitute), (2) a description of the object (metadata), and (3) a description of the commitment made by the NMA regarding the persistence of the object (policy). These services are defined initially to use the HTTP protocol, given the World Wide Web’s pre-eminence among Internet information retrieval systems. When the NMAH is specified, the "ark:" prefix may be replaced with "http://", to produce a valid URL that can gain access to ARK services using an unmodified Web client.

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 for any information resources that accommodate reasonably regular electronic description. This includes digital documents, databases, software, and websites, as well as physical objects (such as books, bones, and statues) 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.

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 that supports name management. The Persistent Uniform Resource Locator [PURL] was a third scheme that has the unique advantage of working with unmodified web browsers. The ARK scheme is a new approach.

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. Rather, 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.

1.1. Three Reasons to Use 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.

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 fleeting life cycle events 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 elusive since the object itself need not include a matching identifier or be transparent 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. Among the recorded characteristics, a checksum (e.g., [MD5]) recorded at the time of last handling may assist automated identification of digital objects (although checksums will require recomputation periodically if extremely persistent objects' bitstreams change as predicted due to inevitable media migration).

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, with persistent identification playing a vital but supporting role. 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 precludes 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.

1.2. Organizing Support for ARKs

Co-location of persistent access and identification services is natural. Any organization undertaking persistent identification and description is in an advantaged position to undertake persistent access, and vice versa. The former task becomes all the easier if the organization controls, owns, or otherwise has clear access to the objects. Similarly, the latter cannot be managed without at least internal support for the former, since collection management activities such as monitoring, acquisition, verification, and change control all require record keeping and accountability. Organizing ARK services under one roof tends to make sense.

ARK support is not for everybody. By requiring specific, revealed commitments to preservation, object access, and description, the bar for providing ARK services is high. 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. Such a business would benefit more from persistence than it would directly support it.

1.3. A Definition of Identifier

Heretofore, persistence discussion has been hampered by a borrowed meaning for "identifier" that emerged as a side effect of defining the Uniform Resource Identifier in [URI]:

> (formerly) An identifier is a sequence of characters with a restricted syntax ... that can act as a reference to something that has identity.

The term works in context, but falters when employed for persistence. Troubling phrases arise, such as,

> "The goal is to create an identifier that does not break."

As defined this kind of identifier "breaks" when it sustains damage to its character sequence, but really what breaks has to do with the identifier's reference role. The following definition is proposed.

> (new definition) An identifier is an *association* between a string (a sequence of characters) and an information resource. That association is made manifest by a record (e.g., a cataloging or other metadata record) that binds the identifier string to a set of identifying resource characteristics.

The identifier (the association) must be vouched for by some sort of record. In the complete absence of any testimony (e.g., metadata) regarding an association, a would-be identifier string is a meaningless sequence of characters. To keep an externally visible but otherwise internal identifier string opaque to 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. If one is lucky an object carries its own identifier as part of itself (e.g., imprinted on the first page), but in processes such as resource discovery and retrieval the typical object is often unwieldy or unavailable (such as when licensing restrictions are in effect). A metadata record that includes the identifier string is the next best thing -- a conveniently manipulable surrogate that can act as both an association "receipt" and "declaration".

It now makes sense to speak of preventing an identifier, as an association, from breaking. Having said that, this document still (ab)uses the terms "ARK" and "identifier" as shorthands to refer to identifier strings, in other words, to sequences of characters. Thus a discussion of ARK syntax refers to a string format, not an association format. The context should make the meaning clear.

2. ARK Anatomy

An ARK is represented by a sequence of characters (a string) that begins with the prefix "ark:". Here is a diagrammed example.

```
         ark:foobar.zaf.org/12025/654xz321
         \__/\____________/ \___/ \______/
          |    (optional)     |      |
  ARK Prefix        |         |    Name (assigned by the NAA)
                    |         |
Name Mapping Authority       Name Assigning Authority
       Hostport (NMAH)        Number (NAAN)
```

The ARK syntax can be summarized,

`ark:`[NMAH]/NAAN/Name

where the NMAH is in brackets to indicate that it is optional.

2.1. The Name Mapping Authority Hostport (NMAH)

After the prefix may appear an optional Name Mapping Authority Hostport (NMAH) that is a temporary address where ARK service requests may be sent. It consists of 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 but one information resource:

```
  ark:foobar.zaf.org/12025/654xz321
ark:sneezy.dopey.com/12025/654xz321
                ark:/12025/654xz321
```

The NMAH makes it easy to derive an identifier that is actionable in today's web browsers (i.e., a URL). This amounts to substituting "http://" for the "ark:" prefix (although in most browsers simply deleting "ark:" works). The first example ARK above thus becomes

```
http://foobar.zaf.org/12025/654xz321
```

The NMAH part is temporary, disposable, and replaceable. Over time the NMAH will likely stop working and have to be replaced with a currently active service provider. This relies on a mapping authority discovery process, of which two alternate methods are outlined in a later section. Meanwhile, a carefully chosen NMAH can be as durable as any Internet domain name, and so may last for a decade or longer. Users should be prepared, however, to refresh the NMAH because the one found in an ARK may have stopped working.

The above method for creating an actionable identifier from an ARK (replacing "ark:" with "http://") is also temporary. Assuming that the reign of [HTTP] in information retrieval will end one day, ARKs will have to be converted into new kinds of actionable identifiers. In any event, if ARKs see widespread use, web browsers would presumably evolve to perform this simple transformation automatically.

2.2. The Name Assigning Authority Number (NAAN)

The next part of 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 number of digits helps reduce the chances of finding a NAAN out of context and confusing it with nearby quantities such as 4-digit dates.

2.3. The Name Part

The final part of the ARK is the Name assigned by the NAA, and it is also required. The Name is a string 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 within HTTP GET requests. Characters may be letters, digits, or any of these seven characters:

```
= @ $ _ * ' #
```

The characters '/', '+', and '?' are reserved and must not be used at this time. A '-' (hyphen) may appear in an ARK, but must be ignored in lexical comparisons. 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].

The creation of names that include linguistically based constructs (having recognizable meaning from natural language) is strongly discouraged if long-term persistence is a naming priority. Such names do not age or travel well. 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 is a relatively safe and easy way to achieve more compact names, although any character repertoire can work if potentially troublesome names will be discarded during a screening process.

2.4. Lexical Equivalence

Hyphens are always ignored in ARKs. Hyphens may be added to an ARK's Name part for readability, or during the formatting and wrapping of text lines, but (as in phone numbers) they are treated as if they were not present. Thus, 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:

```
                     ark:/12025/65-4-xz-321
ark:sneezy.dopey.com/12025/654--xz32-1
                     ark:/12025/654xz321
```

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 ARKs for the purpose of octet-by-octet equality comparison consists of two steps for each ARK. First, any upper case letters in the "ark:" prefix and %-encoded hex digits are converted to lower case. The case of all other letters in the ARK string must be preserved. Then, any NMAH is removed and all hyphens are removed. 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.

2.5. Naming Considerations

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 as part of the ARK.

A valuable technique for provision of persistent objects is to try to have the complete identifier 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.

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 based on naming scheme. Whether a mistakenly re-assigned identifier 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.

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 to outlive brands, corporations, subject classifications, and natural language semantics (e.g., what did the three letters "gay" mean forty, twenty, and two years ago?). 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 identity 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.

Dubious persistence speculation does not make selecting naming strategies any easier. For example, despite rumors to the contrary, there are really 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 would 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. 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 in places like the front matter of a book.

An NAA does not so much create an identifier as create an association. The NAA first draws an identifier 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].

For now, registration of ARK NAAs is in a bootstrapping phase. To register, please read about the mapping authority discovery file in the next section and send email to jak@ckm.ucsf.edu.

4. Finding a Name Mapping Authority

In order to derive an actionable identifier 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.

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.

There is never a requirement that an NAA also run an NMA service, although it seems not an unlikely scenario. Over time NAAs and NMAs would come and go. One NMA would 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 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 the Domain Name System [DNS].

A copy of the current file (at the time of writing) appears in an appendix and is available on the web. A minimal version of the file appears below. Comment lines (lines that begin with ‘#’) explain the format and give 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. 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:   22 February 2001
#     Reload from:   http://ark.nlm.nih.gov/etc/natab
#     Mirrored at:   http://www.ckm.ucsf.edu/people/jak/home/etc/natab
#                    http://....../etc/natab
#     To register:   mailto:jak@ckm.ucsf.edu?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.nlm.nih.gov/naapolicyeg.html for an example).
# All the NMA hostports that service an NAA are listed, one per
# line, indented, after the corresponding NAA line.
#
#   US Library of Congress
12025:  http://www.loc.gov/xxx/naapolicy.html
        foobar.zaf.org
        sneezy.dopey.com
#
#   US National Library of Medicine
12026:  http://www.nlm.nih.gov/xxx/naapolicy.html
        lhc.nlm.nih.gov:8080
        foobar.zaf.org
        sneezy.dopey.com
#
#   US National Agriculture Library
12027:  http://www.nal.gov/xxx/naapolicy.html
        foobar.zaf.gov:80
#
#--- end of data ---
# The enclosed 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+)/);
#     }
# }
# end of file
```

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"     ""   lhc.nlm.nih.gov:8080
 IN NAPTR  0     0   "h"  "ark"     ""   foobar.zaf.org
 IN NAPTR  0     0   "h"  "ark"     ""   sneezy.dopey.com
```

All the fields are held constant for Maptr except for the "flags" 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.

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. 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), and may include the following aspects. One permanence rating framework is given in [NLMPerm].

(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" -- documentation, whether abbreviated or detailed, of any or all corrections, migrations, revisions, etc.

Naming policy for an object includes an historical description of the NAA's (and its successor NAA's) policies regarding differentiation of objects. It may include the following aspects.

(e) "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

(f) "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.

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 Key Mapping Protocol (HKMP)

The HTTP Key Mapping Protocol (HKMP) 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? [HKMP] 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. HKMP is designed so that a person can enter ARK requests directly into the location field of current browser interfaces. Because it runs over HTTP, HKMP 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 an HKMP request (contained within an HTTP request), receives an answer via an HKMP response (contained within an HTTP response), and closes the session. That concludes the connected portion of the protocol.

An HKMP 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 HKMP, prepares and sends an HTTP GET request in the same manner as for a URL. HKMP 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 HKMP 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 three pieces separated from each other by blank lines: the client's piece (lines 1-3), the server's HTTP/HKMP response headers (4-7), and the body of the server's response (8-13). The first and last lines (1 and 13) 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/12025/psbbantu? HTTP/1.1
    C:
    S: HTTP/1.1 200 OK
 5  S: Content-Type: text/plain
    S: HKMP-Status: 0.1 200 OK
    S:
    S: erc:
    S: who:    Lederberg, Joshua
10  S: what:   Studies of Human Families for Genetic Linkage
    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 HKMP, and indicates the HKMP version and a normal return status. The balance of the response (8-11) is the single metadata record that comprises the ARK description service response. The record 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/12025/psbbantu?? HTTP/1.1
    C:
    S: HTTP/1.1 200 OK
 5  S: Content-Type: text/plain
    S: HKMP-Status: 0.1 200 OK
    S:
    S: erc:
    S: who:    Lederberg, Joshua
10  S: what:   Studies of Human Families for Genetic Linkage
    S: when:   1974
    S: where:  http://profiles.nlm.nih.gov/BB/A/N/T/U/_/bbantu.pdf
    S: erc-support:
    S: who:    NIH/NLM/LHNCBC
15  S: what:   Permanent, Unchanging Content
    S: when:   2001 04 21
    S: where:  http://ark.nlm.nih.gov/yy22948
    S: ...  [closes session]
```

Again, a single metadata record (lines 8-17) is returned, but it consists of two segments. The first segment (8-12) 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 (13-17).

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 simple, compact, and printable record designed to hold data associated with an information resource. By design, the ERC is a metadata format that balances the needs for expressive power, very simple machine processing, and direct human manipulation.

A founding principle of the ERC 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 ERC format 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 is based on email and HTTP headers (RFC822) [EMHDRS]. There is a naturalness to its 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 found in 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 expression in an ERC for a born-digital document would be described in the story of the document's release on a web site.

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 one 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
```

Although machine readable, this abbreviated ERC format is neither required nor permitted at this time in HKMP responses.

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.

An ERC segment allows an unlimited number of elements with the same label. Multiple such elements are treated as if they were combined into one element with multiple values. For example, the three elements,

```
who: Bullock, T.H.
who: Achimowicz, J.Z. | Duckrow, R.B. | Spencer, S.S.
who: Iragui-Madoz, V.J.
```

are treated as if all the values were in one element with five-values:

```
who: Bullock, T.H.  | Achimowicz, J.Z. | Duckrow, R.B.
        | Spencer, S.S.  | Iragui-Madoz, V.J.
```

Required ordering is preserved as long as all who's precede all what's, all what's precede all when's, and all when's precede all where's.

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. Such is the case for three out of the four qualifiers appearing in the seven elements below.

```
who/created:    ...
who/published:  ...
who/modified:   ...
when/valid:     ...
when/modified:  ...
when/published: ...
when/created:   ...
```

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 [REG] 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 | v103, i6, p661-678 | 1997 12 00
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.

```
(:unkn)   unknown (e.g., Anonymous)
(:unav)   value unavailable indefinitely
(:none)   never had a value, never will
(:unac)   temporarily inaccessible
(:unap)   not applicable (makes no sense)
(:null)   explicitly empty
(:igno)   element here only to satisfy syntax
(:unas)   value unassigned (e.g., untitled painting)
(:elwr)   value appears elsewhere in record
(:remo)   withheld, suppressed intentionally
```

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 a ws-squeezed block
%}      a non-character that ends a ws-squeezed block
```

One particularly useful construct in ERC element values is the pair of special encoding markers ("%{" and "%}") that indicates a "whitespace-squeezed" 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 take 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
20   001 229 235  9 5 5
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. Stub ERC records have no minimum required elements. They 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 an equivalent URL.

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

A simple expedient that works for now without browser modification is to use a specified-host ARK (one with an NMAH) but without the usual "ark:" prefix (a prefix of "http://" is normally assumed), as in,

```
<a href = "a.b.org/b12345x"> Click Here <a>
```

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

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 (e.g., HTTP, Z39.50) underlying mapping services. 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
Center for Knowledge Management
University of California, San Francisco
530 Parnassus Ave, Box 0840
San Francisco, CA 94143-0840, USA

Fax: +1 415-476-4653
EMail: jak@ckm.ucsf.edu

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

Fax: +1 301-496-0673
EMail: rodgers@nlm.nih.gov

11. References

[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.

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

[ERC] J. Kunze, "Electronic Resource Citations", work in progress.

[HKMP] J. Kunze, "HTTP Key Mapping Protocol", work in progress.

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

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

[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

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

[REG] J. Kunze, "Resource Metadata Vocabulary", work in progress.

[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.

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

12. Appendix: An NLM Prototype ARK Service

The US National Library of Medicine (NLM) 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:   22 February 2001
#     Reload from:   http://ark.nlm.nih.gov/etc/natab
#     Mirrored at:   http://www.ckm.ucsf.edu/people/jak/home/etc/natab
#                    http://....../etc/natab
#     To register:   mailto:jak@ckm.ucsf.edu?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.nlm.nih.gov/naapolicyeg.html for an example).
# All the NMA hostports that service an NAA are listed, one per
# line, indented, after the corresponding NAA line.
#
#   US Library of Congress
12025:  http://www.loc.gov/xxx/naapolicy.html
        foobar.zaf.org
        sneezy.dopey.com
#
#   US National Library of Medicine
12026:  http://www.nlm.nih.gov/xxx/naapolicy.html
        lhc.nlm.nih.gov:8080
        foobar.zaf.org
        sneezy.dopey.com
#
#   US National Agriculture Library
12027:  http://www.nal.gov/xxx/naapolicy.html
        foobar.zaf.gov:80
#
#--- end of data ---
# The enclosed 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+)/);
#     }
# }
# end of file
```

14. Copyright Notice

Copyright (C) The Internet Society (2001). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Expires 8 September 2001

Table of Contents

Status of this Document 1
Abstract 1
1. Introduction 3
1.1. Three Reasons to Use ARKs 3
1.2. Organizing Support for ARKs 4
1.3. A Definition of Identifier 4
2. ARK Anatomy 5
2.1. The Name Mapping Authority Hostport (NMAH) 5
2.2. The Name Assigning Authority Number (NAAN) 6
2.3. The Name Part 6
2.4. Lexical Equivalence 6
2.5. Naming Considerations 7
3. Assigners of ARKs 8
4. Finding a Name Mapping Authority 8
4.1. Looking Up NMAHs in a Globally Accessible File 9
4.2. Looking up NMAHs Distributed via DNS 10
5. Generic ARK Service Definition 12
5.1. Generic ARK Access Service (access, location) 12
5.2. Generic Policy Service (permanence, naming, etc.) 12
5.3. Generic Description Service 13
6. Overview of the HTTP Key Mapping Protocol (HKMP) 13
7. Overview of Electronic Resource Citations (ERCs) 15
7.1. ERC Syntax 16
7.2. ERC Stories 17
7.3. The ERC Anchoring Story 18
7.4. ERC Elements 18
7.5. ERC Element Values 20
7.6. ERC Element Encoding and Dates 22
7.7. ERC Stub Records and Internal Support 23
8. Advice to Web Clients 23
9. Security Considerations 24
10. Authors' Addresses 24
11. References 24
12. Appendix: An NLM Prototype ARK Service 25
13. Appendix: Current ARK Name Authority Table 25
14. Copyright Notice 26