

An Applied ARK Identifier Service Infrastructure

14 March 2007

John Kunze, California Digital Library

What is an Identifier?

- An *identifier* is an association between a string and a thing.
- At last, a non-self-referential definition that...
 - does not fall on its face when applied to the “broken identifier” metaphor
 - comes out and admits that identifiers have no concrete existence, but are only *opinions*
 - like opinions, identifiers may change, diverge, converge, may be mistaken, or ambiguous -- and the best ids and opinions are backed up by facts

But I Want Truth, not Opinion

- There is no truth; there is only (perceived) degree of authority and/or consensus
 - This LCCN is associated with that title in 125 major bibliographic databases worldwide
 - This URL retrieves (is associated with) that object in exactly 1 web server worldwide
 - This CD track exists in 1 database, and is “wrong”
 - When authority/consensus is unclear, steps are taken: check more sources, find experts, etc.
- Don't get upset, get opinions you trust

What Does this Identifier Identify?

- Our best idea of what an identified object *is*:
 - Is our own opinion based on object inspection
 - Most confidence when object wears its identifier
- Often, we inspect metadata that describes it
 - E.g., when we lack the time, rights, or expertise to inspect the object itself
- Required service for high-quality identifiers:
 - Provider X, show me your metadata about Y

Foundation Service: Metadata

- What is an object's metadata?
 - Another object, usually smaller
- Does it need another object identifier?
 - If so, you also need to record relationship
- ARK approach: one identifier, three things:
 - Access to the object, to its metadata, and to a support/commitment statement (metadata)
- ARK: a high-quality, URL-based identifier...
 - and a framework for thinking about identifiers

Identifiers in Action - ARK-style

Two ARKs accessing the “same” thing

<http://cdlib.org/ark:/12025/654xz321>

<http://rutgers.edu/ark:/12025/654xz321>

Access to metadata -- add a ‘?’

<http://cdlib.org/ark:/12025/654xz321?>

Access to support statement -- add ‘??’

<http://cdlib.org/ark:/12025/654xz321??>

- 3 minimal requirements to be an ARK
 - An archive that can’t do all 3 -- trustworthy?
 - Is an ARK persistent? Maybe. Have to *ask*.

Identifier Infrastructure - ARK-style

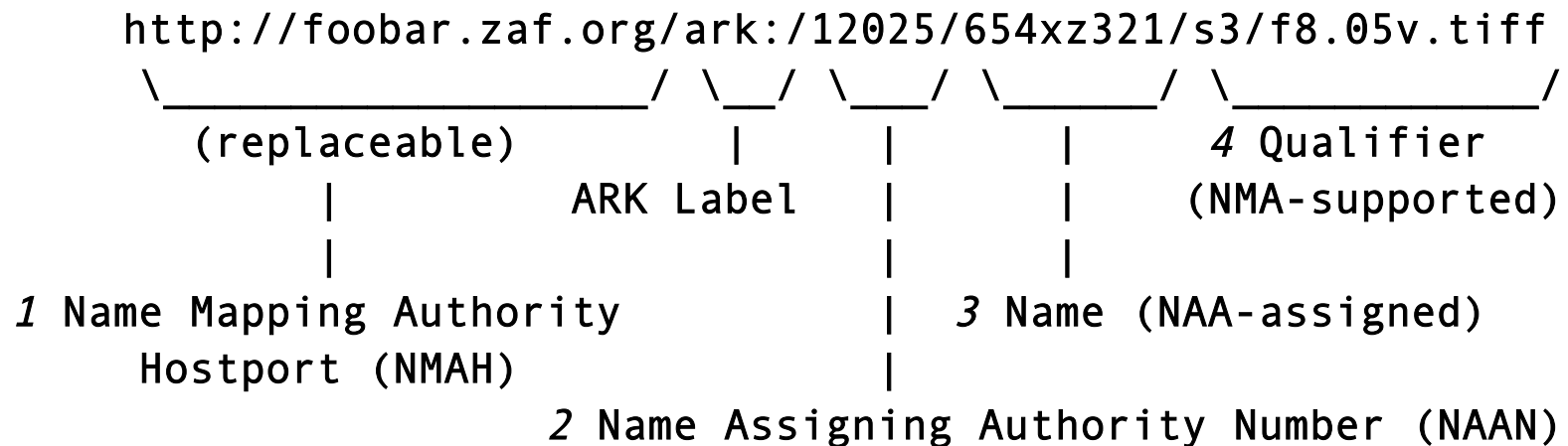
- **Name Assigning Authority (NAA)**: original assigner, with policies on granularity, similarity, and whether ids are ever re-assigned to unrelated objects
- **Name Mapping Authorities (NMAs)**: service providers, in succession or parallel, with policies on their commitment to support/resolve a given id
- Commitments address expected object behavior and expected actions NMA would take in case of loss
- For short-term ids, usually NAA = NMA
- In the long-term, wishes of the NAA become irrelevant; NMAs are object stewards now

Identifier Support - ARK-style

- NMA must respond to ? and ?? incoming
 - Rewrite URLs to invoke lookup of metadata by id
 - Deliver in simple label-colon-value format
 - This is a specific implementation of a generic resolution service independent of HTTP, etc.
- NMA must react to reserved chars . and /
 - Sub-object hierarchy using reserved '/'
 - Variant objects using reserved '.'

ARK Identifier Summary

Instead of one Name Authority: Assigning Authority + Mapping Authorities



- 1 = current service provider; identity inert; replaceable
- 2 = organization that originally assigned the id
- 3 = name originally assigned to the abstract object, often opaque
- 4 = extension disclosing object hierarchy & variants, often non-opaque

Persistence isn't about Identifiers

Thought experiment: imagine the Best Ever Actionable Unique and Terrific Identifiers. These BEAUTI's are just as vulnerable as any URL to

- Funding loss, natural disaster, political and social upheaval
- Processing faults, and errors in human oversight

We don't know much about persistence, but we know some great ways to botch it

Not Botching Persistence

- Use stable hostnames in published URLs; if yours isn't stable, join a consortium
- When possible avoid irritating users; that makes it hard for your descendants
 - The more opaque the id, the better it ages
 - Avoid accidental semantics
 - Get transcription robustness from character set restrictions and check characters
 - Ignore hyphens to neutralize harm done when hyphens are introduced by typesetting

When the Hostname Breaks

- Use low-tech, file lookup (like old /etc/hosts)
- Or use MAPTR algorithm in client or plug-in
 - Resolver discovery using vanilla DNS and script:

```
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 if (! $query || ! $query->answer);
    foreach $rr ($query->answer) {
        next 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 }}}}
```

Opaque Identifier Tools

- What are non-opaque identifier strings? These are strings conceived deliberately to assert some things that are true at the time of assignment
- Opaque identifier strings are best chosen by automated means, such as
 - NOID (nice opaque identifier)
 - **Or** UUID/GUID (universally unique identifier)
 - No need to ask permission or register yourself
 - Looks like something found in nature, but actually it's based on human-maintained IEEE and hardware vendor registries

Nice Opaque Identifiers (NOIDs)

- A noid *minter* is a lightweight database for generating, tracking, and binding unique ids
- The `noid` tool creates minters and accepts commands that operate them
 - Open source, available at www.cpan.org
- Can mint in random or sequential order, with or without a check character guaranteeing against the most common transcription errors
- Anyone can run a noid minter, maintain associations via bindings to arbitrary elements (assertions), and set up a resolver (including rule-based)

Using NOID

- Identifiers minted according to a template:

```
noid dbcreate f5.reedeedk long 13030
```

which produces as first minted id

```
13030/f54x54g11
```

- Noid is scheme-independent
 - Can be used to mint DOIs, URNs, URLs, lotto #s, etc.
 - We (CDL) use it to mint random ARKs with check chars