Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 31, 2015

L. Daigle
TCE

March 2015

Internet Application Identifier Architecture
draft-daigle-appidarch-00.txt

Abstract

This document outlines a general architecture for Internet applications, through the perspective of applications identifiers. It provides a survey of past approaches, drawing out common elements and highlighting common traps and roadblocks.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on August 31, 2015.

Table of Contents

1. Introduction

This document posits a high level architecture of Internet application identifier systems, as well as a survey of IETF efforts dealing in standardization of Internet applications and services using those identifier systems.

Status of this revision: this is a very drafty -00 document. The hope and expectation is that it is enough to stimulate some thought and discussion to flesh out future documents.

2. Basic components of Application Identifier Architecture

There are 3 basic components of an Application Identifier Architecture:

- System
- Identifiers
- Identified

The System is the context in which the identifiers are created, used and in which they are intended to make sense. This is usually transparent, except when identifiers are used outside of this context, causing greater or lesser problems over time. This is discussed in more detail, below.

Identifiers are typically strings of bits or characters. They may have multiple representations.

The concept of what is being Identified is also dependent on the System -- whether it's Internet hosts, services, documents, parts of documents, people or other actors from the physical world, their

representatives in the Internet, etc.

3. Application Identifier Architectures in More Detail

3.1. System

As noted above, the System is the context in which the identifiers make sense. In the Domain Name System, for example, the system initially consisted of the set of hosts attached to the Internet, and it has generalized to the set of addressable Internet services. These are organized into ?domains?, which are operated under the control of a single entity, and individual domains are completely independent of each other.

3.2. Identifiers

Identifiers may identify a thing that exists ? content, service, location ? or is posited to exist. Typical actions on identifiers include:

- "Minting" -- creation of an identifier, usually including association with the identified thing
- Transformation -- changing the bits (characters) of the identifier by some set of rules and/or to conform to some structure; relative or absolute
- Comparison -- of identifiers. Are 2 different identifiers the same? Identifying the same thing? Expressing relationship between things?
- Resolution -- using the identifier to access what it identifies
- Validation -- confirmation that the identifier conforms to the system?s rules (syntax)
- Status check -- has the identifier been minted? Is it active within the system?
- Authentication -- confirmation that the identifier association is valid (as minted)
- Lookup -- some systems support look up ? finding identifier entries based on partial fragments, typically leading characters (bits)
- Search -- some systems support searches for identifiers based on fragments of the related data
- Subscribe -- subscribing to an identifier allows you to get periodic updates as to state of the identifier/identified.

3.3. Identified

Identifiers can be associated with just about any level of concept,

construct, network or software element, or entity in the physical world. The range of possible identified things is generally scoped by the System.

From the perspective of application architectures, there are 4 levels of things that may be identified, and may have individual identifiers:

- Entity/resource -- the thing itself. For example, the published work "Moby Dick"
- Instance -- a specific copy of the thing, e.g., a copy of "Moby Dick".
- Properties -- the characteristics of the thing. The set of properties discussed is generally constrained by the System.
- Relationship -- an identifier may identify something as "part-of" a larger entity.

There are actions that may be performed on the things identified:

- Assign properties -- associate values with properties of identified item
- Get properties
- Intrinsic -- E.g., format, number of words
- Applied -- director's cut
- Publish -- put copy somewhere
- First instance
- Cache/replica
- Get (a copy)
- Any copy
- Specific service
- Closest
- Cheapest
- Authenticate -- confirm (cryptographically) that the resource is genuinely the one expected / related to identifier
- Comparison
- Equivalence

- Send -- a reflection of "get"?

- Subscribe

- Search

4. Survey of existing work

4.1. Domain Name System

The Domain Name System (DNS) was designed to provide identifiers to allow storage and retrieval of (sets of) properties associated with Internet hosts and services ? real and virtual.

DNS identifiers are hierarchical, dot-separated labels, typically expressed as characters. Host names are a subset of domain name identifiers, with some restrictions on the permissible characters.

- "Minting" -- the authority for a domain can create labels within that domain. So-called "synthetic" domain labels are created dynamically.

- Transformation -- relative domain names can be understood within the context of a ?search domain?

- Comparison -- domain names are matched on an octet-by-octet basis (IDNs are not considered here)

- Resolution -- DNS resolution means "DNS lookup" -- using the DNS infrastructure to retrieve resource records associated with the domain name. Resolution can be tailored to retrieve particular types of resource records (e.g., A records, or AAAA records, or MX records)

- Validation -- any string that conforms to DNS syntax may be considered valid.

- Status check -- DNS does not distinguish between "inactive" and "not minted". That is, either every label in the hierarchical domain name is accessible in an authoritative domain server (in which case the domain name is "minted" and "active") or the DNS will return the value that it does not exist. (Not true in DNSSEC?)

- Authentication -- domain names are not authenticated (see below for DNSSEC).

- Lookup -- DNS resolution is lookup of domain names

- Search -- DNS does not support search (wildcards?)

- Subscribe -- N/A

DNS identifies resource records. The resource records are themselves descriptions of Internet hosts, services, and other

information stored in the DNS, but fundamentally the DNS identifier is for resource records.

- Entity/resource -- a set of resource records
- Instance -- copies of resource records may be stored in caching servers; there are no special identifiers to distinguish primary or cached results
- Properties -- N/A
- Relationship -- N/A

There are actions that may be performed on the things identified:

- Assign properties -- resource records have time to live (TTL) and serial numbers included
- Get properties -- parsed as part of the response from the server.
- Publish -- publishing a DNS resource record amounts to updating a DNS zone file with the record.
- Get (a copy) -- resolve the domain name identifier
- Authenticate -- DNSSEC is used to authenticate that the resource records/response received for domain name resolution are as they were published.
- Comparison -- of RRs?
- Send -- N/A
- Search -- N/A

4.2. World Wide Web

The World Wide Web (WWW) is largely defined by the HTTP protocol. "Pages" defined in HTML are the primary design target, though these days much content of varying formats is delivered via the HTTP protocol. For the sake of discussion, we'll say that WWW identifiers are HTTP scheme URIs.

- "Minting" -- typically, HTTP URIs are not composed consciously, so much as assembled practically with components of the domain name hosting the web server and some path components based on how the website is laid out hierarchically (which may or may not relate to

an underlying file structure)

- Transformation -- HTTP URIs may be relative (to current page in the "hierarchy", current domain authority etc)

- Comparison -- HTTP URIs are not inherently comparable except by characterwise comparision or determining relative relationships

- Resolution -- HTTP URIs are resolved by parsing the authority component from the URI, connecting to the server, and requesting the resource associated with the path part of the URI.

- Validation -- any string that conforms to HTTP URI syntax may be considered valid.

- Status check -- HTTP does not distinguish between "inactive" and "not minted". That is, either the HTTP server is available and the resource is found on it (in which case the URI is "minted" and "active") or the server (or path) are not found.

- Authentication -- HTTP URIs are not authenticated (see below for authentication of servers).

- Lookup -- N/A

- Search -- HTTP does not support search (within server?)

- Subscribe -- N/A

HTTP URIs identify "web pages" or "resources".

- Entity/resource -- web content (page)

- Instance -- copies of pages may be stored in caching servers; there are no special identifiers to distinguish primary or cached results

- Properties -- properties may be embedded within the HTML document, but there are no special identifiers to query/retrieve properties; as part of the HTTP protocol, capabilities may be negotiated

- Relationship -- relative URIs (?)

There are actions that may be performed on the things identified:

- Assign properties -- the web server may assign properties to a web page.

- Get properties --parsed as part of the response from the server.

- Publish -- publishing a web page is done on the backend, out of band of the WWW system

- Get (a copy) -- resolve the URI

- Authenticate -- certs are used, within HTTP, to authenticate the server is empowered to operate for a given domain name. Individual pages are not authenticated.

- Comparison -- many web pages look alike -- but there is no inherent way to claim two web pages (documents) are "the same".

- Send -- N/A

- Search -- within the WWW there is no support for search (all search is achieved as an external system).

4.3. Classic URIs

The advent of the WWW heralded a burst of development of standards for applications and content on the Internet. Much work was done to elaborate a general system of identifiers, supporting a broad range of application needs -- Uniform Resource Identifiers in the large, encompassing Locators (identifiers of Internet "location"), Names (persistent, location-independent identifiers for resources), Characteristics (metadata about resources), Agents (for composing actions).

In the large, the classic perspective on URIs was that they would identify all resources (documents, services, media, components, classes, parameters etc) that would be referenced within Internet protocols.

- "Minting" -- dependent on the URI scheme. The HTTP URI scheme is outlined above as a dynamic URI creation example. Some (namespaces of) URNs require more explicit minting of identifiers to be used in URNs (e.g., ISBN URNs).

- Transformation -- dependent on the URI scheme. URNs were intended to be (authoritatively) transformed into URLs identifying the location of the desired resource at a given point in time.

- Comparison -- Scheme-dependent -- there is no URI-wide support for comparing URIs (except byte-wise equality).

- Resolution -- Scheme-dependent. Some URI schemes do not have Internet-based resolution capabilities.

- Validation -- any string that conforms to URI syntax may be considered valid. Individual schemes may include validation services (e.g., out of band lookup services, built-in checksums, etc).

- Status check -- Generally, URIs do not distinguish between

"inactive" and "not minted". That is, successful resolution

implies minted, unsuccessful resolution is ambiguous. Individual schemes may provide more refined methods of confirmation of status (SIP URIs?)

- Authentication -- There is no URI-wide support for URI authentication.

- Lookup -- N/A

- Search -- URIs are not inherently searchable

- Subscribe -- N/A

Classically, URIs identify Internet "resources". However, URIs have been found in contexts that are disjoint from the Internet (e.g., XML component identification). "Resource" is deliberately general -- could be documents, services, etc. Each URI scheme refines its scope of intended resources.

Entity/resource -- typically Internet content or service, but see comment above.

Instance -- Most URI schemes do not support identification of instances. However, URNs are intended to identify locations of multiple instances of a given resource.

Properties -- URIs may identify URCs (Uniform Resource Characteristics) an articulation of the properties of a given resource. (URCs were never finally standardized).

Relationship -- relative URIs (?)

There are actions that may be performed on the things identified:

- Assign properties -- publish a URC

- Get properties -- retrieve a URC associated with the URI.

- Publish -- scheme-dependent

- Get (a copy) -- resolve the URI

- Authenticate -- certs are used, within some URI schemes, to authenticate the server is empowered to operate for a given domain name. Individual resources are not authenticated. (Unless you have a URC with a checksum?)

- Comparison -- many resources look alike -- but there is no inherent way to claim two resources are "the same".

- Send -- N/A

- Search -- within the URI space there is no support for search (all search is achieved as an external system).

4.4. IP addresses

To be added... the interesting thing about IP addresses is considering the context in which they are defined, and then contrasting that with the places they turn up.

5. Common design choices and challenges

In creating a system, there are important common design choices that need to be made. Sometimes the answer is implicit within the overall design constraints of the system. Other times, considerable effort is required to refine specifications and make appropriate choices. As noted, these are common design questions. This is an area where understanding of previous systems? design discussions can be particularly helpful (in order not to repeat them needlessly).

5.1. Identifiers

In defining identifiers for a system, is the intention that they:

- Name something -- the identifier will be associated with one entity (or instance of an entity), wherever that entity may be located.

- Locate something -- identify the location of an entity (at some point in time).

- Are Smart or dumb identifiers -- "smart" identifiers have structures that can be parsed to determine something about the thing identified (e.g., domain in which it is stored); "dumb" identifiers are opaque and must be resolved within the system.

- Have uniqueness -- is the identifier/resource binding unique?

- Have scope -- is the uniqueness (or other properties) only maintained within some limited scope, or is it global?

- Permanent -- what is the expected level of permanence of the identifier's relevance (the ability to use it, the binding between the identifier and the identified resource). Or, are they transient identifiers?

5.2. Identified

The specifics of the identified resource need to be defined, as well.

- Instances -- can there be multiple instances of a single resource? How can they be distinguished and/or how can two instances be equated. This is important if one needs to be able to cache

instances or otherwise validate "local" copies.

- Scope of applicability -- what constitutes a "resource" in this system?

6. Issues in (mis)using identifiers

Things like using IP addressed out of context of the routing system ? assumptions about uniqueness and volatility may be improper.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

This document is about considering applications systems. Security is important to applications, but is not specifically called out here.

9. References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://xml.resource.org/public/rfc/html/rfc2119.html>.

Author's Address

Leslie Daigle
ThinkingCat Enterprises
Leesburg, VA 20176
US

Email: ldaigle@thinkingcat.com