Application and Packaging Pointer (app) URI scheme
draft-soilandreyes-app-04

Abstract

   This specification proposes the Application and Packaging Pointer URI
   scheme "app".

   app URIs can be used to consume or reference hypermedia resources
   bundled inside a file archive or an application package, as well as
   to resolve URIs for archive resources within a programmatic
   framework.

   This URI scheme provides mechanisms to generate a unique base URI to
   represent the root of the archive, so that relative URI references in
   a bundled resource can be resolved within the archive without having
   to extract the archive content on the local file system.

   An app URI can be used for purposes of isolation (e.g. when consuming
   multiple archives), security constraints (avoiding "climb out" from
   the archive), or for externally identiyfing sub-resources referenced
   by hypermedia formats.

Table of Contents

1. Introduction

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in [RFC2119].

   For the purpose of this specification, an *archive* is a collection
   of sub-resources addressable by name or path. This definition covers
   typical archive file formats like ".zip" or "tar.gz" and derived
   "+zip" media types [RFC6839], but also non-file resource packages
   like an LDP Container [W3C.REC-ldp-20150226], an installed Web App
   [W3C.WD-appmanifest-20180118], or a BagIt folder structure
   [I-D.draft-kunze-bagit-14].

   For brevity, the term _archive_ is used throughout this
   specification, although from the above it can also mean a
   container, _application_ or _package_.

2. Background

   Mobile and Web Applications ("apps") may bundle resources such as
   stylesheets with relative URI references to scripts, images and
   fonts. Resolving such resources within URI handling frameworks may
   require generating absolute URIs and applying Same-Origin [RFC6454]
   security policies separately for each app.

   Applications that are accessing resources bundled inside an archive
   (e.g. "zip" or "tar.gz" file) can struggle to consume hypermedia
   content types that use relative URI references [RFC3986] such as
   "../css/", as it is challenging to determine the base URI in a
   consistent fashion.

   Frequently the archive must be unpacked locally to synthesize base
   URIs like "file:///tmp/a1b27ae03865/" to represent the root of the
   archive. Such URIs are temporary, might not be globally unique, and
   could be vulnerable to attacks such as "climbing out" of the root
   directory.

   An archive containing multiple HTML or Linked Data resources, such as
   in a BagIt archive [I-D.draft-kunze-bagit-14], may be using relative
   URIs to cross-reference constituent files.

   Consumptions of archives might be performed in memory or through a
   common framework, abstracting away any local file location.

   Consumption of an archive with a consistent base URL should be
   possible no matter from which location it was retrieved, or on which
   device it is inspected.

When consuming multiple archives from untrusted sources it would be
beneficial to have a Same Origin policy [RFC6454] so that relative
hyperlinks can't escape the particular archive.

The "file:" URI scheme [RFC8089] can be ill-suited for purposes such
as above, where a location-independent URI scheme is more flexible,
secure and globally unique.

3. Scheme syntax

The "app" URI scheme follows the [RFC3986] syntax for hierarchical
URIs according to the following productions:

URI   = scheme ":" app-specific ["#" fragment]

scheme   = "app"

app-specific = "//" app-authority [path-absolute] ["?" query]

The "app-authority" component provides a unique identifier for the
opened archive. See Section 3.1 for details.

The "path-absolute" component provides the absolute path of a
resource (e.g. a file or directory) within the archive. See
Section 3.2 for details.

The "query" component MAY be used, but its semantics is undefined by
this specification.

The "fragment" component MAY be used by implementations according to
[RFC3986] and the implied media type [RFC2046] of the resource at the
path. This specification does not specify how to determine the media
type.

3.1. Authority

The purpose of the "authority" component in an app URI is to build a
unique base URI for a particular archive. The authority is NOT
intended to be resolvable without former knowledge of the archive.

The authority of an app URI MUST be valid according to these
productions:

app-authority = uuid | ni | name | authority
uuid   = "uuid," UUID
ni   = "ni," alg-val
name   = "name," reg-name

1. The prefix "uuid," combines with the "UUID" production as defined
    in [RFC4122], e.g. "uuid,2a47c495-ac70-4ed1-850b-8800a57618cf"

2. The prefix "ni," combines with the "alg-val" production as
    defined in [RFC6920], e.g. "ni,sha-
    256;JCS7yveugE3UaZiHCs1XpRVfSHaewxAKka0o5q2osg8"

3. The prefix "name," combines with the "reg-name" production as
    defined in [RFC3986], e.g. "name,app.example.com".

4. The production "authority" matches its definition in [RFC3986].
    As this necessarily also match the above prefixed productions,
    those should be considered first before falling back to this
    production.

3.2. Path

The "path-absolute" component, if present, MUST match the production
in [RFC3986] and provide the absolute path of a resource (e.g. a file
or directory) within the archive.

Archive media types vary in constraints and possibilities on how to
express paths, however implementations SHOULD use "/" as path
separator for nested folders and files.

It is RECOMMENDED to include the trailing "/" if it is known the path
represents a directory.

4. Scheme semantics

This specification does not constrain what format might constitute an
archive, and neither does it require that the archive is
retrievable as a single bytestream or file.

Examples of retrievable archive media types include "application/
zip", "application/vnd.android.package-archive", "application/x-tar",
"application/x-gtar" and "application/x-7z-compressed".

Examples of non-file archives include an LDP Container
[W3C.REC-ldp-20150226], an installed Web App
[W3C.WD-appmanifest-20180118], or a BagIt folder structure
[I-D.draft-kunze-bagit-14].

4.1. Authority semantics

The _authority_ component identifies the archive itself.

Implementations MAY assume that two app URIs with the same authority
component relate to resources within the same archive, subject to
limitations explained in this section.

The authority prefix, if present, helps to inform consumers what
uniqueness constraints have been used when identifying the archive,
without necessarily providing access to the archive.

1. If the prefix is "uuid," followed by a UUID [RFC4122], this
    indicates a unique archive identity.

2. If the prefix is "uuid," followed by a v4 UUID [RFC4122], this
    indicate uniqueness based on a random number generator.
    Implementations creating random-based authorities SHOULD generate
    the v4 random UUID using a suitable random number generator
    [RFC4086].

3. If the prefix is "uuid," followed by a v5 name-based UUID
    [RFC4122], this indicates uniqueness based on an existing archive
    location, typically an URL. Implementations creating location-
    based authorities from an archive's URL SHOULD generate the v5
    UUID using the URL namespace "6ba7b811-9dad-
    11d1-80b4-00c04fd430c8" and the particular URL (see [RFC4122]
    section 4.3). Note that while implementations cannot resolve
    which location was used, they can confirm the name-based UUID if
    the location is otherwise known.

4. If the prefix is "ni," this indicates a unique archive identity
    based on a hashing of the archive's bytestream or content.
    Implementations can assume that resources within an "ni" app URIs
    remains static, although the implementation may use content
    negotiation or similar transformations. The checksum MUST be
    expressed according to [RFC6920]'s "alg-val" production.
    Implementations creating hash-based authorities from an archive's
    bytestream SHOULD use the "sha-256" without truncation.

5. If the prefix is "name," this indicates that the authority is an
    application or package name, typically as installed on a device
    or system. Implementations SHOULD assume that an unrecognized
    "name" authority is only unique within a particular installation,
    but MAY assume further uniqueness guarantees for names under
    their control. It is RECOMMENDED that implementations creating
    name-based authorities use DNS names under their control, for
    instance an app installed as "app.example.com" can make an
    authority "name,app.example.com" to refer to its packaged
    resources, or "name,foo.app.example.com" to refer to a "foo"
    container distributed across all installations.

The uniqueness properties are unspecified for app URIs which
authority do not match any of the prefixes defined in this
specification.

4.2. Path semantics

The _path_ component of an app URI identify individual resources
within a particular archive, typically a _directory_ or _file_.

o If the _path_ is "/" - e.g. "app://uuid,833ebda2-f9a8-4462-b74a-
   4fcdc1a02d22/" - then the app URI represent the archive itself,
   typically represented as a root directory or collection.

o If the path ends with "/" then the path represents a directory or
   collection.

The app URIs can be used for uniquely identifying the resources
independent of the location of the archive, such as within an
information system.

Assuming an appropriate resolution mechanism which have knowledge of
the corresponding archive, an app URI can also be used for
resolution.

Some archive formats might permit resources with the same (duplicate)
path, in which case it is undefined from this specification which
particular entry is described.

4.3. Resolution protocol

This specification do not define the protocol to resolve resources
according to the app URI scheme. For instance, one implementation
might rewrite app URIs to localized paths in a temporary directory,
while another implementation might use an embedded HTTP server.

It is envisioned that an implementation will have extracted or opened
an archive in advance, and assigned it an appropriate authority
according to Section 3.1. Such an implementation can then resolve
app URIs programmatically, e.g. by using in-memory access or mapping
paths to the extracted archive on the local file system.

Implementations that support resolving app URIs SHOULD:

1. Fail with the equivalent of _Not Found_ if the authority is
    unknown.

2. Fail with the equivalent of _Gone_ if the authority is known, but
    the content of the archive is no longer available.

   3. Fail with the equivalent of _Not Found_ if the path does not map
       to a file or directory within the archive.

   4. Return the corresponding (potentially uncompressed) bytestream if
       the path maps to a file within the archive.

   5. Return an appropriate directory listing if the path maps to a
       directory within the archive.

   6. Return an appropriate directory listing of the archive's root
       directory if the path is "/".

   Not all archive formats or implementations will have the concept of a
   directory listing, in which case the implementation MAY fail such
   resolutions with the equivalent of "Not Implemented".

   It is not undefined by this specification how an implementation can
   determine the media type of a file within an archive. This could be
   expressed in secondary resources (such as a manifest), be determined
   by file extensions or magic bytes.

   The media type "text/uri-list" [RFC2483] MAY be used to represent a
   directory listing, in which case it SHOULD contain only URIs that
   start with the app URI of the directory.

   Some archive formats might support resources which are neither
   directories nor regular files (e.g. device files, symbolic links).
   This specification does not define the semantics of attempting to
   resolve such resources.

   This specification does not define how to change an archive or its
   content using app URIs.

4.4. Resolving from a .well-known endpoint

   If the "authority" component of an app URI matches the "alg-val"
   production, an application MAY attempt to resolve the authority from
   any ".well-known/ni/" endpoint [RFC5785] as specified in [RFC6920]
   section 4, in order to retrieve the complete archive. Applications
   SHOULD verify the checksum of the retrieved archive before resolving
   the individual path.

5. Encoding considerations

   The productions for "UUID" and "alg-val" are restricted to URI safe
   ASCII and should not require any encoding considerations.

Care should be taken to %-encode the directory and file segments of
"path-absolute" according to [RFC3986] (for URIs) or [RFC3987] (for
IRIs).

When used as part an IRI, paths SHOULD be expressed using
international Unicode characters instead of %-encoding as ASCII.

Not all archive formats have an explicit character encoding specified
for their paths. If no such information is available for the archive
format, implementations MAY assume that the path component is encoded
with UTF-8 [RFC2279].

Some archive formats have case-insensitive paths, in which cases it
is RECOMMENDED to preserve the casing as expressed in the archive.

6. Interoperability considerations

As multiple authorities are possible for the same archive
(Section 3.1), and path interpretation might vary, there can be
interoperability challenges when exchanging app URIs between
implementations. Some considerations:

1. Two implementations describe the same archive (e.g. stored in the
    same local file path), but using different random-based UUID
    authorities. The implementations may need to detect equality of
    the two UUIDs out of band.

2. Two implementations describe an archive retrieved from the same
    URL, with the same location-based UUID authority, but retrieved
    at different times. The implementations might disagree about the
    content of the archive.

3. Two implementations describe an archive retrieved from the same
    URL, with the same location-based UUID authority, but retrieved
    using different content negotiation resulting in different
    archive representations. The implementations may disagree about
    path encoding, file name casing or hierarchy.

4. Two implementations describe the same archive bytestream using
    the hash-based authority, but they have used two different hash
    algorithms. The implementations may need to negotiate to a
    common hash algorithm.

5. Two implementations access the same archive, which contain file
    paths with Unicode characters, but extracted to two different
    file systems. Limitations and conventions for file names in the
    local file system (such as Unicode normalization, case

insensitivity, total path length) may result in the
implementations having inconsistent or inaccessible paths.

7. Security Considerations

As when handling any content, extra care should be taken when
consuming archives and app URIs from unknown sources.

An archive could contain compressed files that expand to fill all
available disk space.

A maliciously crafted archive could contain paths with characters
(e.g. backspace) which could make an app URI invalid or misleading if
used unescaped.

A maliciously crafted archive could contain paths (e.g. combined
Unicode sequences) that cause the app URI to be very long, causing
issues in information systems propagating said URI.

An archive might contain symbolic links that, if extracted to a local
file system, might address files outside the archive's directory
structure. Implementations SHOULD detect such links and prevent
outside access.

An maliciously crafted app URI might contain "../" path segments,
which if naively converted to a "file:///" URI might address files
outside the archive's directory structure. Implementations SHOULD
perform Path Segment Normalization [RFC3986] before converting app
URIs.

In particular for IRIs, an archive might contain multiple paths with
similar-looking characters or with different Unicode combine
sequences, which could be used to mislead users.

An URI hyperlink might use or guess an app URI authority to attempt
to climb into a different archive for malicious purposes.
Applications SHOULD employ Same Orgin policy [RFC6454] checks if
resolving cross-references is not desired.

While a UUID or hash-based authority provide some level of
information hiding of an archive's origin, this should not be relied
upon for access control or anonymisation. Implementors should keep
in mind that such authority components in many cases can be
predictably generated by third-parties, for instance using dictionary
attacks.

8. IANA Considerations

This specification requests that IANA registers the following URI
scheme according to the provisions of [RFC7595].

Scheme name: app

Status: provisional

Applications/protocols that use this protocol: Hypermedia-consuming
application that handle archives or packages.

Contact: Stian Soiland-Reyes stain@apache.org [1]

Change controller: Stian Soiland-Reyes

9. References

9.1. Normative References

[RFC2046]   Freed, N. and N. Borenstein, "Multipurpose Internet Mail
            Extensions (MIME) Part Two: Media Types", RFC 2046,
            DOI 10.17487/RFC2046, November 1996,
            <https://www.rfc-editor.org/info/rfc2046>.

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119,
            DOI 10.17487/RFC2119, March 1997,
            <https://www.rfc-editor.org/info/rfc2119>.

[RFC2279]   Yergeau, F., "UTF-8, a transformation format of ISO
            10646", RFC 2279, DOI 10.17487/RFC2279, January 1998,
            <https://www.rfc-editor.org/info/rfc2279>.

[RFC2483]   Mealling, M. and R. Daniel, "URI Resolution Services
            Necessary for URN Resolution", RFC 2483,
            DOI 10.17487/RFC2483, January 1999,
            <https://www.rfc-editor.org/info/rfc2483>.

[RFC3986]   Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
            Resource Identifier (URI): Generic Syntax", STD 66,
            RFC 3986, DOI 10.17487/RFC3986, January 2005,
            <https://www.rfc-editor.org/info/rfc3986>.

[RFC3987]   Duerst, M. and M. Suignard, "Internationalized Resource
            Identifiers (IRIs)", RFC 3987, DOI 10.17487/RFC3987,
            January 2005, <https://www.rfc-editor.org/info/rfc3987>.

   [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker,
              "Randomness Requirements for Security", BCP 106, RFC 4086,
              DOI 10.17487/RFC4086, June 2005,
              <https://www.rfc-editor.org/info/rfc4086>.

   [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally
              Unique IDentifier (UUID) URN Namespace", RFC 4122,
              DOI 10.17487/RFC4122, July 2005,
              <https://www.rfc-editor.org/info/rfc4122>.

   [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known
              Uniform Resource Identifiers (URIs)", RFC 5785,
              DOI 10.17487/RFC5785, April 2010,
              <https://www.rfc-editor.org/info/rfc5785>.

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

   [RFC6839] Hansen, T. and A. Melnikov, "Additional Media Type
              Structured Syntax Suffixes", RFC 6839,
              DOI 10.17487/RFC6839, January 2013,
              <https://www.rfc-editor.org/info/rfc6839>.

   [RFC6920] Farrell, S., Kutscher, D., Dannewitz, C., Ohlman, B.,
              Keranen, A., and P. Hallam-Baker, "Naming Things with
              Hashes", RFC 6920, DOI 10.17487/RFC6920, April 2013,
              <https://www.rfc-editor.org/info/rfc6920>.

   [RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines
              and Registration Procedures for URI Schemes", BCP 35,
              RFC 7595, DOI 10.17487/RFC7595, June 2015,
              <https://www.rfc-editor.org/info/rfc7595>.

   [RFC8089] Kerwin, M., "The "file" URI Scheme", RFC 8089,
              DOI 10.17487/RFC8089, February 2017,
              <https://www.rfc-editor.org/info/rfc8089>.

9.2. Informative References

   [ApacheTaverna]
              "Apache Taverna (incubating)", January 2018,
              <https://taverna.incubator.apache.org/>.

   [I-D.draft-kunze-bagit-14]
              Kunze, J., Littman, J., Madden, L., Summers, E., Boyko,
              A., and B. Vargas, "The BagIt File Packaging Format
              (V0.97)", draft-kunze-bagit-14 (work in progress), October
              2016.

   [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data
              Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006,
              <https://www.rfc-editor.org/info/rfc4648>.

   [RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M.,
              and D. Orchard, "URI Template", RFC 6570,
              DOI 10.17487/RFC6570, March 2012,
              <https://www.rfc-editor.org/info/rfc6570>.

   [ROBundle]
              Soiland-Reyes, S., Gamble, M., and R. Haines, "Research
              Object Bundle 1.0", Zenodo report,
              DOI 10.5281/zenodo.12586, November 2014,
              <https://w3id.org/bundle/>.

   [W3C.NOTE-app-uri-20150723]
              Caceres, M., "The app: URL Scheme", World Wide Web
              Consortium NOTE NOTE-app-uri-20150723, July 2015,
              <http://www.w3.org/TR/2015/NOTE-app-uri-20150723>.

   [W3C.NOTE-widgets-uri-20120313]
              Caceres, M., "Widget URI scheme", World Wide Web
              Consortium NOTE NOTE-widgets-uri-20120313, March 2012,
              <http://www.w3.org/TR/2012/NOTE-widgets-uri-20120313>.

   [W3C.REC-ldp-20150226]
              Speicher, S., Arwe, J., and A. Malhotra, "Linked Data
              Platform 1.0", World Wide Web Consortium Recommendation
              REC-ldp-20150226, February 2015,
              <http://www.w3.org/TR/2015/REC-ldp-20150226>.

   [W3C.WD-appmanifest-20180118]
              Caceres, M., Christiansen, K., Lamouri, M., Kostiainen,
              A., and R. Dolin, "Web App Manifest", World Wide Web
              Consortium WD WD-appmanifest-20180118, January 2018,
              <https://www.w3.org/TR/2018/WD-appmanifest-20180118>.

9.3. URIs

   [1] mailto:stain@apache.org

[Appendix A](). Examples

[A.1](). Sharing using app names

   A photo gallery application on a mobile device uses app URIs for
   navigation between its UI states. The gallery is secured so that
   other applications can't normally access its photos.

   The application is installed as the package name
   "gallery.example.org", making the corresponding name-based app URI:

   app://name,gallery.example.org/

   A user is at the application state which shows the newest photos as
   thumbnails:

   app://name,gallery.example.org/photos/?New

   The user selects a photo, rendered with metadata overlaid:

   app://name,gallery.example.org/photos/137

   The user requests to "share" the photo, selecting
   "messaging.example.com" which uses the common URI framework on the
   device.

   The photo gallery registers with the device's app framework that the
   chosen "messaging.example.com" gets read permission to its
   "/photos/137" resource.

   The sharing function returns a URI Template [[RFC6570]()]:

   app://name,messaging.example.com/share{;uri}{;redirect}

   Filling in the template, the gallery requests to pop up:

   app://name,messaging.example.com/share
     ;uri=app://gallery.example.org/photos/137
     ;redirect=app://gallery.example.org/photos/%3fNew

   The app framework checks its registration for "messaging.example.com"
   and finds the installed messaging application. It performs
   permission checks that other apps are allowed to navigate to its
   "/share" state.

   The messaging app is launched and navigates to its "sharing" UI,
   asking the user for a caption.

The messaging app requests the app framework to retrieve
"app://name,gallery.example.org/photos/137" using content negotiation
for an "image/jpeg" representation.

The app framework finds the installed photo gallery
"gallery.example.org", and confirms the read permission.

The photo gallery application returns a JPEG representation after
retrieving the photo from its internal store.

After the messaging app has completed sharing the picture bytestream,
it request the UI framework to navigate to:

app://name,gallery.example.org/photos/?New

The UI returns to the original view in the photo gallery.

If the messaging app had attempted to _retrieve_ the app URI

app://name,gallery.example.org/photos/?New

then it would be rejected by the app framework as permission was not
granted.

However, if such access had been granted, the gallery could return a
"text/uri-list" of the newest photos:

app://name,gallery.example.org/photos/137
app://name,gallery.example.org/photos/138
app://name,gallery.example.org/photos/139

This examples show that although an app URI represents a resource, it
can have different representations or UI states for different apps.

A.2. Sandboxing

An document store application has received a file "document.tar.gz"
which content will be checked for consistency.

For sandboxing purposes it generates a UUID v4 "32a423d6-52ab-47e3-
a9cd-54f418a48571" using a pseudo-random generator. The app base URI
is thus "app://uuid,32a423d6-52ab-47e3-a9cd-54f418a48571/"

The archive contains the files:

o "./doc.html" which links to "css/base.css"

o "./css/base.css" which links to "../fonts/Coolie.woff"

   o "./fonts/Coolie.woff"

   The application generates the corresponding app URIs and uses those
   for URI resolutions to list resources and their hyperlinks:

```
 app://uuid,32a423d6-52ab-47e3-a9cd-54f418a48571/doc.html
   -> app://uuid,32a423d6-52ab-47e3-a9cd-54f418a48571/css/base.css
 app://uuid,32a423d6-52ab-47e3-a9cd-54f418a48571/css/base.css
   -> app://uuid,32a423d6-52ab-47e3-a9cd-54f418a48571/fonts/Coolie.woff
 app://uuid,32a423d6-52ab-47e3-a9cd-54f418a48571/fonts/Coolie.woff
```

   The application is now confident that all hyperlinked files are
   indeed present in the archive. In its database it notes which
   "tar.gz" file corresponds to UUID "32a423d6-52ab-47e3-a9cd-
   54f418a48571".

   If the application had encountered a malicious hyperlink
   "../../../outside.txt" it would first resolve it to the absolute URI
   "app://uuid,32a423d6-52ab-47e3-a9cd-54f418a48571/outside.txt" and
   conclude from the "Not Found" error that the path "/outside.txt" was
   not present in the archive.

[A.3](#). Origin-based

   A web crawler is about to index the content of the URL
   "http://example.com/data.zip" and need to generate absolute URIs as
   it continues crawling inside the individual resources of the archive.

   The application generates a UUID v5 based on the URL namespace
   "6ba7b811-9dad-11d1-80b4-00c04fd430c8" and the URL to the zip file:

```
   >>> uuid.uuid5(uuid.NAMESPACE_URL, "http://example.com/data.zip")
   UUID('b7749d0b-0e47-5fc4-999d-f154abe68065')
```

   Thus the location-based app URI for indexing the ZIP content is

```
   app://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/
```

   Listing all directories and files in the ZIP, the crawler finds the
   URIs:

```
   app://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/
   app://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/pics/
   app://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/pics/flower.jpeg
```

   When the application encounters "http://example.com/data.zip" some
   time later it can recalculate the same base app URI. This time the

ZIP file has been modified upstream and the crawler finds
additionally:

app://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/pics/cloud.jpeg

If files had been removed from the updated ZIP file the crawler can
simply remove those from its database, as it used the same app base
URI as in last crawl.

A.4. Hash-based

An application where users can upload software distributions for
virus checking needs to avoid duplication as users tend to upload
"foo-1.2.tar" multiple times.

The application calculates the "sha-256" checksum of the uploaded
file to be in hexadecimal:

17edf80f84d478e7c6d2c7a5cfb4442910e8e1778f91ec0f79062d8cbdef42cd

The "base64url" encoding [RFC4648] of the binary version of the
checksum is:

F-34D4TUeOfG0selz7REKRDo4XePkewPeQYtjL3vQs0

The corresponding "alg-val" authority is thus:

sha-256;F-34D4TUeOfG0selz7REKRDo4XePkewPeQYtjL3vQs0

From this the hash base app URL is:

app://ni,sha-256;F-34D4TUeOfG0selz7REKRDo4XePkewPeQYtjL3vQs0/

The crawler finds that its virus database already contain entries
for:

app://ni,sha-256;F-34D4TUeOfG0selz7REKRDo4XePkewPeQYtjL3vQs0/bin/evil

and flags the upload as malicious without having to scan it again.

A.5. Archives that are not files

An application is relating BagIt archives [I-D.draft-kunze-bagit-14]
on a shared file system, using structured folders and manifests
rather than individual archive files.

The BagIt payload manifest "/gfs/bags/scan15/manifest-md5.txt" lists
the files:

```
49afbd86a1ca9f34b677a3f09655eae9 data/27613-h/q172.png
408ad21d50cef31da4df6d9ed81b01a7 data/27613-h/q172.txt
```

The application generates a random UUID v4 "ff2d5a82-7142-4d3f-b8cc-3e662d6de756" which it adds to the bag metadata file "/gfs/bags/scan15/bag-info.txt"

External-Identifier: ff2d5a82-7142-4d3f-b8cc-3e662d6de756

It then generates app URIs for the files listed in the manifest:

```
app://uuid,ff2d5a82-7142-4d3f-b8cc-3e662d6de756/data/27613-h/q172.png
app://uuid,ff2d5a82-7142-4d3f-b8cc-3e662d6de756/data/27613-h/q172.txt
```

When a different application on the same shared file system encounter these app URIs, it can match them to the correct bag folder by inspecting the "External-Identifier" metadata.

A.6. Linked Data containers which are not on the web

An application exposes in-memory objects of an Address Book as a Linked Data Platform container [W3C.REC-ldp-20150226], but addressing the container using app URIs instead of http to avoid network exposure.

The app URIs are used in conjuction with a generic LDP client library (developed for http), but connected to the application's URI resolution mechanism.

The application generates a new random UUID v4 "12f89f9c-e6ca-4032-ae73-46b68c2b415a" for the address book, and provides the corresponding app URI to the LDP client:

```
app://uuid,12f89f9c-e6ca-4032-ae73-46b68c2b415a/
```

The LDP client resolves the container with content negotiation for the "text/turtle" media type, and receives:

```
@base <app://uuid,12f89f9c-e6ca-4032-ae73-46b68c2b415a/>.
@prefix ldp: <http://www.w3.org/ns/ldp#>.
@prefix dcterms: <http://purl.org/dc/terms/>.

<app://uuid,12f89f9c-e6ca-4032-ae73-46b68c2b415a/>
  a ldp:BasicContainer;
  dcterms:title "Address book";
  ldp:contains <contact1>, <contact2>.
```

The LDP client resolves the relative URIs to retrieve each of the
contacts:

```
app://uuid,12f89f9c-e6ca-4032-ae73-46b68c2b415a/contact1
app://uuid,12f89f9c-e6ca-4032-ae73-46b68c2b415a/contact2
```

A.7. Resolution of packaged resources

A virtual file system driver on a mobile operating system has mounted
several packaged applications for resolving common resources. An
application requests the rendering framework to resolve a picture
from "app://uuid,eb1edec9-d2eb-4736-a875-eb97b37c690e/img/logo.png"
to show it within a user interface.

The framework first checks that the authority "uuid,eb1edec9-d2eb-
4736-a875-eb97b37c690e" is valid to access according to the Same
Origin policies or permissions of the running application. It then
matches the authority to the corresponding application package.

The framework resolves "/img/logo.png" from within that package, and
returns an image buffer it already had cached in memory.

Appendix B. History

This specification proposes the URI scheme "app", which was
originally proposed by [W3C.NOTE-app-uri-20150723] but never
registered with IANA. That W3C Note evolved from
[W3C.NOTE-widgets-uri-20120313] which proposed the URI scheme
"widget".

Neither W3C Notes progressed further as Recommendation track
documents.

While the focus of those W3C Notes was to specify how to resolve
resources from within a packaged application, this specification
generalize the "app" URI scheme to support referencing and
identifying resources within any archive, and de-emphasize the
retrieval mechanism.

For compatibility with existing adaptations of the "app" URI scheme,
e.g. [ROBundle] and [ApacheTaverna], this specification reuse the
same scheme name and remains compatible with the intentions of
[W3C.NOTE-app-uri-20150723], but renames "app" to mean "Application
and Packaging Pointer" instead of "Application".

Authors' Addresses

    Stian Soiland-Reyes
    The University of Manchester
    Oxford Road
    Manchester
    United Kingdom

    Email: stain@apache.org

    Marcos Caceres
    Mozilla Corporation

    Email: marcos@marcosc.com