Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 21, 2018

S. Soiland-Reyes
The University of Manchester

January 17, 2018

The app URI scheme
draft-soilandreyes-app-00

Abstract

This Internet-Draft proposes the "app" URI scheme for the Archive and Packaging Protocol.

app URIs can be used to consume or reference hypermedia resources bundled inside a file archive or a mobile 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 in other hypermedia formats.

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].

This Internet-Draft will expire on July 21, 2018.

Copyright Notice

Table of Contents

1. Background

Applications that are accessing resources bundled inside a file archive (e.g. zip or tar.gz) can struggle to consume hypermedia

content types that use relative URI references [RFC3986], 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 fluctual, might not be globally unique, and could be vulnerable to attacks such as "climbing out" of the root directory.

Mobile and Web applications that are distributed as packages may bundle resources such as stylesheets with relative URI references to images and fonts.

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.

2. Scheme syntax

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

```
appURI    =  "app://" app-authority [ path-absolute ]
               [ "?" query ] [ "#" fragment ]
```

The "app-authority" component provides a unique identifier for the opened archive. See Section 2.1 for details.

The "path-absolute" component provides the absolute path of a resource (e.g. a file or directory) within the archive. See Section 2.2 for details.

The semantics of the "query" component is undefined by this Internet-Draft. Implementations SHOULD NOT generate a query component for app URIs.

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

2.1. 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 this production:

```
app-authority    = UUID | alg-val | authority
```

The "UUID" production match its definition in [RFC4122], e.g. "2a47c495-ac70-4ed1-850b-8800a57618cf"

The "alg-val" production match its definition in [RFC6920], e.g. "sha-256;JCS7yveugE3UaZiHCs1XpRVfSHaewxAKka0o5q2osg8"

The "authority" production match its definition in [RFC3986], e.g. "example.com". As this production necessarily also match the "UUID" and "alg-val" productions, consumers of app URIs should attempt to match those first. While [RFC7320] section 2.2 says an extension may not "define the structure or the semantics for URI authorities", extensions of this Internet-Draft *are* permitted to do so, if using a DNS domain name under their control. For instance, a vendor owning "example.com" may specify that "{OID}" in "{OID}.oid.example.com" has special semantics.

The choice of authority depends on the purpose of the app URI within the implementation. Below are some recommendations:

1. _Sandboxing_, when independently interpreting resources in an archive, the authority SHOULD be a UUID v4 [RFC4122] created with a suitable random number generator [RFC4086]. This ensures with high probablity that the app base URI is globally unique. An application MAY choose to reuse a previously assigned UUID that is associated with the archive.

2. _Location-based_, for referencing resources in an archive accessed at a particular URL, the authority SHOULD be generated

as a name-based UUID v5 [RFC4122]; that is based on the SHA1 concatination of the URL namespace "6ba7b811-9dad-11d1-80b4-00c04fd430c8" (as UUID bytes) and the ASCII bytes of the particular URL. It is NOT RECOMMENDED to use this approach with a file URI [RFC8089] without a fully qualified "host" name.

3. _Hash-based_, for referencing resources in an archive as a particular bytestream, independent of its location, the authority SHOULD be a checksum of the archive bytes. The checksum MUST be expressed according to [RFC6920]'s "alg-val" production, and SHOULD use the "sha-256" algorithm. It is NOT RECOMMENDED to use truncated hash methods.

The generic "authority" production MAY be used for extensions if the above mechanisms are not suitable. Care should be taken so that the custom "authority" do not match the "UUID" nor "alg-val" productions.

2.2. Path

The "path-absolute" component 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 flexibilities of how to express paths. Here we assume an archive generally consists of a single root directory, which can contain multiple directories and files at arbitrary nesting levels.

Paths SHOULD be expressed using "/" as the directory separator. The below productions are from [RFC3986]:

```
path-absolute = "/" [ segment-nz *( "/" segment ) ]
segment       = *pchar
segment-nz    = 1*pchar
```

In an app URI, each intermediate "segment" (or "segment-nz") represent a directory name, while the last segment represent either a directory or file name.

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

3. Scheme semantics

This Internet-Draft does not constrain what particular format might constitute an _archive_, and neither does it require that the archive is retrievable as a single bytestream or file. Examples of archive media types include "application/zip", "application/

vnd.android.package-archive", "application/x-tar", "application/x-gtar" and "application/x-7z-compressed".

The _authority_ component identifies the archive file.

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

- If the _path_ is missing/empty - e.g. "app://833ebda2-f9a8-4462-b74a-4fcdc1a02d22" - then the app URI represent the whole archive file.

- If the _path_ is "/" - e.g. "app://833ebda2-f9a8-4462-b74a-4fcdc1a02d22/" - then the app URI represent the root directory of the archive.

- If the path ends with "/" then the path represents a directory in the archive

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.

3.1. Resolution protocol

This Internet-Draft do not specify directly the protocol to resolve resources according to the app URI scheme. For instance, one implementation might rewrite app URIs to localized "file:///" 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 2.1. 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 "/"

7. Return the archive file if the path component is missing/empty.

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

It is not specified in this Internet-Draft how an implementation can determine the media type of a file within an archive. This may 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 Internet-Draft does not specify the semantics of attempting to resolve such resources.

This Internet-Draft does not specify how to change an archive or its content using app URIs.

3.2. 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.

4. Encoding considerations

The production for "UUID" and "alg-val" are restricted to 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 media types 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 media types are case-insensitive, in which cases it is RECOMMENDED to preserve the casing as expressed in the archive.

5. Interoperability considerations

As multiple authorities are possible (Section 2.1), there could 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 v4 UUIDs. 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 v5 UUIDs, 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 v5 UUIDs, 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 "alg-val" production, but they have used two different hash algorithms. The implementations may need to negotiate to a common hash algorithm.

5. An implementation describe an archive using the "alg-val" production, but a second implementation concurrently modifies the archive's content. The first implementation may need to detect changes to the archive or verify the checksum at the end of its operations.

6. Two implementations might have different views of the content of the same archive if the format permits multiple entries with the same path. Care should be taken to follow the convention and specification of the particular archive format.

7. Two implementations that access the same archive which contain file paths with Unicode characters, but they extract to two different file systems. Limitations and conventions for file names in the local file system (e.g. Unicode normalization, case insensitivity, total path length) may result in the implementations having inconsistent or inaccessible paths.

6. 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.

An maliciously crafted app URI might contain "../" segments, which if naively converted to a "file:///" URI might address files outside the archive's directory structure.

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

7. IANA Considerations

This Internet-Draft contains the Provisional IANA registration of the app URI scheme according to [RFC7595].

Scheme name: app

Status: provisional

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

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

Change controller: Stian Soiland-Reyes

8. References

8.1. 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>.

[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>.

[RFC7320] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 7320, DOI 10.17487/RFC7320, July 2014, <https://www.rfc-editor.org/info/rfc7320>.

[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>.

8.2. Informative References

[CWLViewer]
Robinson, M., Soiland-Reyes, S., and M. Crusoe, "Common-Workflow-Language/CWLviewer: CWL Viewer", Zenodo Software, DOI 10.5281/zenodo.823534, August 2017, <https://view.commonwl.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>.

[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>.

8.3. URIs

[1] mailto:stain@apache.org

Appendix A. Examples

A.1. 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://32a423d6-52ab-47e3-a9cd-54f418a48571/"

The archive contains the files:

- "./doc.html" which links to "css/base.css"
- "./css/base.css" which links to "../fonts/Coolie.woff"
- "./fonts/Coolie.woff"

The application generates the corresponding app URIs and uses those for URI resolutions:

- app://32a423d6-52ab-47e3-a9cd-54f418a48571/doc.html links to app://32a423d6-52ab-47e3-a9cd-54f418a48571/css/base.css

- app://32a423d6-52ab-47e3-a9cd-54f418a48571/css/base.css` links to app://32a423d6-52ab-47e3-a9cd-54f418a48571/fonts/Coolie.woff

- app://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 ZIP file corresponds to "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://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.2. 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 base app URI is "app://b7749d0b-0e47-5fc4-999d-f154abe68065/" for indexing the ZIP content, after which the crawler finds:

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

- app://b7749d0b-0e47-5fc4-999d-f154abe68065/pics/

- app://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:

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

If files had been removed from the updated ZIP file this would be trivial for the crawler to clear from its database, as it used the same base URI as in last crawl.

A.3. 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 "17edf80f84d478e7c6d2c7a5cfb4442910e8e1778f91ec0f79062d8cbdef42cd" in hexadecimal. 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" meaning the base app URL is "app://sha-256;F-34D4TUeOfG0selz7REKRDo4XePkewPeQYtjL3vQs0/"

The crawler finds that it's virus database already contain entries for:

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

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

A.4. 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/images/q172.png
408ad21d50cef31da4df6d9ed81b01a7 data/27613-h/images/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://ff2d5a82-7142-4d3f-b8cc-3e662d6de756/data/27613-h/images/q172.png
app://ff2d5a82-7142-4d3f-b8cc-3e662d6de756/data/27613-h/images/q172.txt
```

A.5. Resolution of packaged resources

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

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

Appendix B. History

This Internet-Draft 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 did progress further as Recommendation track documents.

While the focus of W3C Notes was to specify how to resolve resources from within a packaged application, this Internet-Draft 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 [CWLViewer], this Internet-Draft reuse the same scheme name and remains compatible with the intentions of [W3C.NOTE-app-uri-20150723].

Author's Address

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

Email: stain@apache.org