Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 9, 2018

S. Soiland-Reyes
The University of Manchester
M. Caceres
Mozilla Corporation

February 05, 2018

The Archive and Package (arcp) URI scheme
draft-soilandreyes-arcp-03

Abstract

This specification define the Archive and Package URI scheme "arcp".

arcp 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 arcp 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.

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 https://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 9, 2018.

Table of Contents

1. Introduction

Mobile and Web Applications may bundle resources such as stylesheets with _relative URI references_ [RFC3986] (4.2 [1]) to scripts, images and fonts. Resolving and parsing such resources within URI handling frameworks may require generating absolute URIs and applying Same-Origin [RFC6454] security policies separately for each app.

Software that is 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 such as "../css/", as it is challenging to establishing the _base URI_ [RFC3986] (5.1 [2]) in a consistent fashion.

Frequently the archive might be unpacked locally, implying 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, making it challenging to index or annotate such resources.

Consumption of an archive with a consistent base URL should be possible no matter from which location it was retrieved, on which device it is inspected, and with which mechanism the archive is accessed (e.g. virtual file system).

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, while a location-independent URI scheme can be more flexible, secure and globally unique.

This specification define the Archive and Package URI scheme "arcp" as an alternative to addressing resources within an archive, application or package.

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_, _aggregation_ or _package_.

The main purpose of arcp URIs is to provide consistent identifiers as absolute URIs for nested resources. This specification does not define a new network protocol, however it suggests an abstract resolution protocol that implementations can apply using existing protocols or programming frameworks.

2. Requirements Language

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].

3. Scheme syntax

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

```
arcp-URI       = arcp-scheme ":" arcp-specific [ "#" fragment ]

arcp-scheme    = "arcp"

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

The "arcp-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 arcp URI is to build a unique identifier for a particular archive. The authority is NOT intended to be resolvable without former knowledge of the archive.

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

```
arcp-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". For arcp IRIs [RFC3987], its "ireg-name" production applies instead of "reg-name".

4. The production "authority" matches its definition in [RFC3986], or "iauthority" for arcp IRIs [RFC3987]. 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], or "ipath-absolute" for arcp IRIs [RFC3987]. This 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 that 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 arcp 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. Applications MAY assume that the corresponding "urn:uuid:" URI identifies the archive.

2. If the prefix is "uuid," followed by a v4 UUID [RFC4122] (4.4 [3]), 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] (4.3 [4]), this indicates uniqueness based on an existing archive location, typically an URL.
   Implementations creating location-based authorities SHOULD generate the v5 UUID using the URL namespace "6ba7b811-9dad-11d1-80b4-00c04fd430c8" and an retrievable archive URL. 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 MAY assume that resources within an "ni" arcp URIs remains static, although the implementation may use content negotiation or similar transformations.
   The checksum MUST be expressed according to the "alg-val" production in [RFC6920] (3 [5]). Implementations creating hash-based authorities from an archive's bytestream SHOULD use the hash method "sha-256" without truncation. Implementations MAY assume that the corresponding "ni:" URI identifies the archive.

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 unrecognised "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 its dynamic container of "foo" resources.

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

4.2. Path semantics

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

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

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

The arcp URIs can be used for uniquely identifying resources within an archive, such as in an information system considering multiple archives.

Assuming an appropriate mechanism which have knowledge of the corresponding archive, an arcp 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 a network protocol to resolve resources according to the arcp URI scheme. For instance, one implementation might rewrite arcp 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 accessed an archive in advance, and assigned it an appropriate authority according to Section 3.1. Such an implementation can then resolve arcp URIs, e.g. by using in-memory archive access or mapping arcp paths to the the local file system.

Implementations that support resolving arcp 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 resource 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 "/".

Implementations MAY support other ways to resolve arcp URIs, e.g. query parameters or content negotiation.

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 with the arcp URI of the directory as a common base.

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 arcp URIs.

4.4. Resolving from a .well-known endpoint

If the "authority" component of an arcp URI matches the "alg-val" production, an application MAY assume corresponding "ni:///" or "nih:" URIs [RFC6920] identify the archive bytestream or content.

Applications MAY attempt to retrieve the corresponding archive from any ".well-known/ni/" endpoint [RFC5785] as specified in [RFC6920] (4 [6]). Applications SHOULD verify the checksum of the retrieved archive before resolving individual arcp paths.

5. Encoding considerations

The productions for "uuid" and "ni" are restricted to URI safe ASCII and should not require any encoding considerations.

When arcp is used in IRIs [RFC3987], the "name" production permit Unicode characters corresponding to its "ireg-name" production.

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

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 arcp 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 formats. 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 arcp URIs from unknown sources.

Archives might contain malicious or inappropriate content or file paths.

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 arcp URI invalid or misleading if used unescaped.

A maliciously crafted archive could contain paths with character combinations (e.g. combined Unicode sequences, text orientation change) that cause the arcp URI to be very long or disruptive when rendered in an user interface.

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 arcp 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 arcp 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 arcp 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: arcp

Status: provisional

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

Contact: Stian Soiland-Reyes stain@apache.org [7]

Change controller: Stian Soiland-Reyes

9. References

9.1. Normative References

[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>.

[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>.

9.2. Informative References

[FirefoxOS] Mozilla Firefox, "Firefox OS security overview", MDN Mozilla Developer Network Web Docs, February 2017, <https://developer.mozilla.org/en-US/docs/Archive/B2G_OS/Security/Security_model#Packaged_Apps>.

[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.

[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>.

[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>.

[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>.

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

[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] https://tools.ietf.org/html/rfc3986#section-4.2

[2] https://tools.ietf.org/html/rfc3986#section-5.1

[3] https://tools.ietf.org/html/rfc4122#section-4.4

[4] https://tools.ietf.org/html/rfc4122#section-4.3

[5] https://tools.ietf.org/search/rfc6920#section-3

[6] https://tools.ietf.org/html/rfc6920#section-4

[7] mailto:stain@apache.org

Appendix A. Examples

A.1. Sandboxing base URI

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 arcp base URI is thus:

arcp://uuid,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/Foo.woff"
- "./fonts/Foo.woff"

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

```
arcp://uuid,32a423d6-52ab-47e3-a9cd-54f418a48571/doc.html
  -> arcp://uuid,32a423d6-52ab-47e3-a9cd-54f418a48571/css/base.css
arcp://uuid,32a423d6-52ab-47e3-a9cd-54f418a48571/css/base.css
  -> arcp://uuid,32a423d6-52ab-47e3-a9cd-54f418a48571/fonts/Foo.woff
arcp://uuid,32a423d6-52ab-47e3-a9cd-54f418a48571/fonts/Foo.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 <arcp://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.2. Location-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 arcp URI for indexing the ZIP content is

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

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

```
arcp://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/
arcp://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/pics/
arcp://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 arcp URI. This time the ZIP file has been modified upstream and the crawler finds additionally:

```
arcp://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 arcp base URI as in last crawl.

A.3. Hash-based

A repository where users can annotate content of open source software distributions needs to avoid duplication, as users tend to upload "foo-1.2.tar" multiple times.

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

```
7f83b1657ff1fc53b92dc18148a1d65dfc2d4b1fa3d677284addd200126d9069
```

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

```
f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk
```

The corresponding "alg-val" authority [RFC6920] is thus:

```
sha-256;f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk
```

From this the hash-based arcp base URL is:

arcp://ni,sha-256;f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk/

The repository adds annotations for detected source code files within the archive.

A client is browsing the annotations and discovers:

arcp://ni,sha-256;f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk/src/luhn.c

The client constructs the corresponding "ni" URI [RFC6920]:

ni:///sha-256;f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk/

To retrieve the archive from "repo.example.com", the client resolve the corresponding ".well-known" URI [RFC5785]:

http://repo.example.com/.well-known/
    ni/sha-256/f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk/

After the client verifies the corresponding "sha-256" checksum it reads the path "/src/luhn.c" from the retrieved archive.

A.4. Archives that are not files

An application is storing BagIt archives [I-D.draft-kunze-bagit-14] on a shared file system, using structured "bag" 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" and adds the corresponding "urn:uuid" UUID to the bag metadata file "/gfs/bags/scan15/bag-info.txt"

External-Identifier: urn:uuid:ff2d5a82-7142-4d3f-b8cc-3e662d6de756/

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

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

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

A.5. 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 arcp URIs instead of http to avoid network exposure.

The arcp 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 arcp URI to the LDP client:

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

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

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

<arcp://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:

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

A.6. 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 to show it within a user interface:

```
<img src="arcp://name,app.example.com/img/logo.png"
         alt="App" />
```

The framework finds the corresponding application package, installed as "app.example.com". It then checks that the authority "name,app.example.com" is valid to access according to the Same Origin policies or permissions of the running application.

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

A.7. Sharing using app names

A photo gallery application on a mobile device uses arcp 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 app name "gallery.example.org" as the vendor controls "example.org", making the corresponding name-based arcp URI:

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

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

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

The user selects a photo, rendered with metadata overlaid:

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

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

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

The sharing function returns a URI Template [RFC6570]:

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

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

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

The arcp 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 arcp framework to retrieve the "uri" <arcp://name,gallery.example.org/photos/137> using content negotiation for an "image/jpeg" representation.

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

The photo gallery application returns a scaled down 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 the "redirect" state:

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

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

This example shows that although an arcp URI represents a resource, it can have different representations or views in different apps.

Appendix B. Acknowledgements

This specification is inspired by two original URI scheme proposals from W3C, "app" from [W3C.NOTE-app-uri-20150723] and "widget" from [W3C.NOTE-widgets-uri-20120313].

The "app" URI scheme was used by packaged web apps in Mozilla's Firefox OS [FirefoxOS] and to identify resources in Research Object Bundles [ROBundle], however the W3C Notes did not progress further as W3C Recommendation track documents, and their URI schemes were never formally registered with IANA.

While the focus of the previous proposals was to specify how to resolve resources from within a packaged application, this specification generalize the URI scheme to support referencing and identifying resources within any archive, package or application, and adding flexibility for how resources can be resolved.

The authors would like to thank Graham Klyne, Carsten Bormann, Roy T. Fielding, S Moonesamy, Julian Reschke and Frank Ellermann for valuable feedback and suggestions.

Authors' Addresses

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

Email: stain@apache.org
URI: http://orcid.org/0000-0001-9842-9718

Marcos Caceres
Mozilla Corporation

Email: marcos@marcosc.com
URI: http://marcosc.com/