Network Working Group
Internet Draft
Intended status: Standards Track
Expires: September 12, 2012

Paul E. Jones
Gonzalo Salgueiro
Cisco Systems
Joseph Smarr
Google

March 12, 2012

Webfinger
draft-jones-appsawg-webfinger-01.txt

Abstract

This specification defines the Webfinger protocol. Webfinger may be used to discover information about people on the Internet, such as a person's personal profile address, identity service, telephone number, or preferred avatar. Webfinger may also be used to learn information about objects on the network, such as the amount of toner in a printer or the physical location of a server.

Status of this Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2012.

Table of Contents

1. Introduction

There is a utility found on UNIX systems called "finger" [14] that allows a person to access information about another person. The information being queried might be on a computer anywhere in the world. The information returned via "finger" is simply a plain text file that contains unstructured information provided by the queried user.

Webfinger borrows the concept of the legacy finger protocol, but introduces a very different approach to sharing information. Rather than returning a simple unstructured text file, Webfinger uses structured documents that contain link relations. These link relations point to information a user or entity on the Internet wishes to expose. For a person, the kinds of information that might be exposed include a personal profile address, identity service, telephone number, or preferred avatar. Webfinger may also be used to

learn information about objects on the network, such as the amount of toner in a printer or the physical location of a server.

Information returned via Webfinger might be for direct human consumption (e.g., another user's phone number) or it might be used by systems to help carry out some operation (e.g., facilitate logging into a web site by determining a user's identification service).

2. Terminology

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 RFC 2119 [1].

"SHOULD", "SHOULD NOT", "RECOMMENDED", and "NOT RECOMMENDED" are appropriate when valid exceptions to a general requirement are known to exist or appear to exist, and it is infeasible or impractical to enumerate all of them. However, they should not be interpreted as permitting implementors to fail to implement the general requirement when such failure would result in interoperability failure.

Webfinger makes heavy use of "Link Relations". Briefly, a Link Relation is an attribute and value pair used on the Internet wherein the attribute identifies the type of link to which the associated value refers. In Hypertext Transfer Protocol (HTTP) [2] and Web Linking [3], the attribute is a "rel" and the value is an "href".

3. Example Uses of Webfinger

In this section, we describe just a few sample uses for Webfinger and show what the protocol looks like. This is not an exhaustive list of possible uses and the entire section should be considered non-normative. The list of potential use cases is virtually unlimited since a user can share any kind of machine-consumable information via Webfinger.

3.1. Locating a User's Blog

Assume you receive an email from Bob and he refers to something he posted on his blog, but you do not know where Bob's blog is located. It would be simple to discover the address of Bob's blog if he makes that information available via Webfinger.

Let's assume your email client discovers that blog automatically for you. When receive the message from Bob (bob@example.com), your email client performs the following steps behind the scenes.

First, it tries to get the host metadata [9] information for the domain example.com. It does this by issuing the following HTTPS query to example.com:

```
GET /.well-known/host-meta HTTP/1.1
Host: example.com
```

The server replies with an XRD [8] document:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/xrd+xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<XRD xmlns="http://docs.oasis-open.org/ns/xri/xrd-1.0">
  <Link rel="lrdd"
        type="application/xrd+xml"
        template="https://example.com/lrdd/?uri={uri}"/>
</XRD>
```

The client then processes the received XRD in accordance with the Web Host Metadata [9] procedures. The client will see the LRDD link relation and issue a query with the user's account URI [5]. (The Account URI is discussed in Section 4.2.) The query might look like this:

```
GET /lrdd/?uri=acct%3Abob%40example.com HTTP/1.1
Host: example.com
```

The server might then respond with a message like this:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/xrd+xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<XRD xmlns="http://docs.oasis-open.org/ns/xri/xrd-1.0">
  <Subject>acct:bob@example.com</Subject>
  <Link rel="http://webfinger.net/rel/avatar"
        href="http://www.example.com/~bob/bob.jpg"/>
  <Link rel="http://webfinger.net/rel/profile-page"
        href="http://www.example.com/~bob/"/>
  <Link rel="blog"
        href="http://blogs.example.com/bob/"/>
</XRD>
```

The email client might take note of the "blog" link relation in the above XRD document that refers to Bob's blog. This URL would then be presented to you so that you could then visit his blog.

The email client might also note that Bob has published an avatar link relation and use that picture to represent Bob inside the email client.

3.2. Retrieving a Person's Contact Information

Assume you have Alice in your address book, but her phone number appears to be invalid. You could use Webfinger to find her current phone number and update your address book.

Let's assume you have a web-based address book that you wish to update. When you instruct the address book to pull Alice's current contact information, the address book might issue a query like this to get host metadata information for example.com:

```
GET /.well-known/host-meta.json HTTP/1.1
Host: example.com
```

Note the address book is looking for a JSON [4] representation, whereas we used XML in the previous example.

The server might reply with something like this:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=UTF-8

{
  "links" :
  [
    {
      "rel" : "lrdd",
      "type" : "application/json",
      "template" :
        "https://example.com/lrdd/?format=json&uri={uri}"
    }
  ]
}
```

The client processes the response as described in RFC 6415 [9]. It will process the LRDD link relation using Alice's account URI by issuing this query:

```
GET /lrdd/?format=json&uri=acct%3Aalice%40example.com HTTP/1.1
Host: example.com
```

The server might return a response like this:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=UTF-8

{
  "subject" : "acct:alice@example.com",
  "links" :
  [
    {
      "rel" : "http://webfinger.net/rel/avatar",
      "href" : "http://example.com/~alice/alice.jpg"
    },
    {
      "rel" : "vcard",
      "href" : "http://example.com/~alice/alice.vcf"
    }
  ]
}
```

With this response, the address book might see the vcard [16] link relation and use that file to offer you updated contact information.

3.3. Simplifying the Login Process

OpenID (http://www.openid.net) is great for allowing users to log into a web site, though one criticism is that it is challenging for users to remember the URI they are assigned. Webfinger can help address this issue by allowing users to use user@domain-style addresses. Using a user's account URI, a web site can perform a query to discover the associated OpenID identifier for a user.

Let's assume Carol is trying to use OpenID to log into a blog. The blog server might issue the following query to get the host metadata information:

```
GET /.well-known/host-meta.json HTTP/1.1
Host: example.com
```

The response that comes back is similar to the previous example:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=UTF-8
{
  "links" :
  [
    {
      "rel" : "lrdd",
      "type" : "application/json",
```

```
        "template" :
          "https://example.com/lrdd/?format=json&uri={uri}"
      }
    ]
  }
```

The blog server processes the response as described in RFC 6415. It will process the LRDD link relation using Carol's account URI by issuing this query:

```
GET /lrdd/?format=json&uri=acct%3Acarol%40example.com HTTP/1.1
```

The server might return a response like this:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=UTF-8

{
  "subject" : "acct:carol@example.com",
  "links" :
  [
    {
      "rel" : "http://webfinger.net/rel/avatar",
      "href" : "http://example.com/~alice/alice.jpg"
    },
    {
      "rel" : " http://specs.openid.net/auth/2.0/provider ",
      "href" : "https://openid.example.com/carol"
    }
  ]
}
```

At this point, the blog server knows that Carol's OpenID identifier is https://openid.example.com/carol and could then proceed with the login process as usual.

3.4. Retrieving Device Information

While the examples thus far have been focused on information about humans, Webfinger does not limit queries to only those that use the account URI scheme. Let's suppose there are devices on the network like printers and you would like to check the current toner level for a particular printer identified via the URI device:p1.example.com.

Following the procedures similar to those above, a query may be issued to get link relations specific to this URI like this:

```
GET /lrdd/?format=json&uri=device%3Ap1.example.com HTTP/1.1
Host: example.com
```

The link relations that are returned may be quite different than those for human users. Perhaps we may see a response like this:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=UTF-8

{
  "subject" : "device:p1.example.com",
  "links" :
  [
    {
      "rel" : "tipsi",
      "href" : "http://192.168.1.5/npap/"
    }
  ]
}
```

While this example is entirely fictitious, you can imagine that perhaps the Transport Independent, Printer/System Interface [18] may be enhanced with a web interface that allows a device that understands the TIP/SI web interface specification to query the printer for toner levels.

4. Webfinger Protocol

Webfinger does not actually introduce a new protocol, per se. Rather, it builds upon the existing Web Host Metadata [9] specification and leverages the Cross-Origin Resource Sharing (CORS) [7] specification.

4.1. Performing a Webfinger Query

The first step a client must perform in executing a Webfinger query is to query for the host metadata using HTTPS or HTTP. The procedures are defined in the Web Host Metadata [9] specification.

Webfinger clients MUST locate the LRDD link relation, if present, and perform a query for that link relation, if present. All other link templates found must be processed to form a complete resource descriptor. The processing rules in Section 4.2 of RFC 6415 MUST be followed.

Webfinger servers MUST accept requests for both XRD [8] and JRD [9] documents. The default representation returned by the server MUST be an XRD document, but a JRD document MUST be returned if the client

explicitly requests it by using /.well-known/host-meta.json or includes an Accept header in the HTTP request with a type of "application/json" [4].

If the client requests a JRD document when querying for host metadata, the Webfinger server can assume that the client will want a JRD documents when querying the LRDD resource. As such, when the Webfinger server returns a JRD document containing host metadata it should include a URI for an LRDD resource that can return a JRD document and MAY include a URI for an LRDD resource that will return an XRD document.

If the client queries the LRDD resource and provides a URI for which the server has no information, the server MUST return a 404 status code. Likewise, any query to a URI in the resource descriptor that is unknown to the server should result in the server returning a 404 status code.

4.2. The Web Host Metadata "resource" Parameter

In addition to the normal processing logic for processing host metadata information, Webfinger defines the "resource" parameter for querying for host metadata and returning all of the link relations from LRDD and other resource-specific link templates in a single query. This resource essentially pushes the work to the server to form a complete resource descriptor for the specified resource.

Note that support for the "resource" parameter is optional, but strongly recommended for improved performance. If a server does not implement the "resource" parameter, then the server's host metadata processing logic remains unchanged from RFC 6415.

To utilize the host-meta "resource" parameter, a Webfinger client issues a request to /.well-known/host-meta or /.well-known/host-meta.json as usual, but then appends a "resource" parameter as shown in this example:

```
GET /.well-known/host-meta.json?resource=\
                      acct%3Abob%40example.com HTTP/1.1
Host: example.com
```

Note that the "\" character shown above is to indicate that the line breaks at this point and continues on the next line. This was shown only to avoid line wrapping in this document and is not a part of the HTTP protocol.

When processing this request, the Webfinger server MUST

* Return a 404 status code if the URI provided in the resource parameter is unknown to the server; and

* Set the "Subject" returned in the response to the value of the "resource" parameter if the URI provided in the resource parameter is known to the server

The Webfinger client can verify support for the "resource" parameter by checking the value of the Subject returned in the response. If the Subject matches the value of the "resource" parameter, then the "resource" parameter is supported by the server.

For illustrative purposes, the following is an example usage of the "resource" parameter that aligns with the example in Section 1.1.1 of RFC 6415. The Webfinger client would issue this request:

```
GET /.well-known/host-meta.json?resource=\
                        http%3A%2F%2Fexample.com%2Fxy HTTP/1.1
Host: example.com
```

The Webfinger server would reply with this response:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=UTF-8

{
  "subject" : "http://example.com/xy",
  "properties" :
  {
    "http://spec.example.net/color" : "red"
  },
  "links" :
  [
    {
      "rel" : "hub",
      "href" : "http://example.com/hub"
    },
    {
      "rel" : "hub",
      "href" : "http://example.com/another/hub"
    },
    {
      "rel" : "author",
      "href" : "http://example.com/john"
    },
    {
      "rel" : "author",
```

```
           "template" : "http://example.com/author?\
```

```
                              q=http%3A%2F%2Fexample.com%2Fxy"
          }
        ]
      }
```

5. The "acct" URI

The Web Host Metadata specification [9] allows for any kind of resource to be queried, but Webfinger defines a specific type of resource in order to query information about a human user.

Specifically, Webfinger uses the "acct" URI to refer to a human user's account on the Internet.

5.1. Using the "acct" URI

The "acct" URI takes a familiar form in looking like an email address. However, the account URI is not an email address and should not be mistaken for one. Quite often, the account URI minus the "acct:" scheme prefix may be exactly the same as the user's email address.

A user MUST NOT be required to enter the "acct" URI scheme name along with his account identifier into any Webfinger client. Rather, the Webfinger client MUST accept identifiers that are void of the "acct:" portion of the identifier. Composing a properly formatted "acct" URI is the responsibility of the Webfinger client.

A user MAY provide a fully-specified "acct" URI.

5.2. Syntax of "acct" URI

The "acct" URI syntax is defined here in Augmented Backus-Naur Form (ABNF) [6] and borrows syntax elements from RFC 3986 [5]:

```
acctURI      =  "acct:" userpart "@" domainpart
userpart     =  1*( unreserved / pct-encoded )
domainpart   =  domainlabel 1*( "." domainlabel)
domainlabel  =  alphanum / alphanum *( alphanum / "-" ) alphanum
alphanum     =  ALPHA / DIGIT
```

The "acct" URI scheme allows any character from the Unicode [11] character set encoded as a UTF-8 [19] string that is then percent-encoded as necessary into valid ASCII [20]. Characters in the domainpart must be encoded to support internationalized domain names (IDNs) [12].

Characters in the userpart or domainpart that are not unreserved must be percent-encoded when used in a protocol or document that only

supports or requires ASCII. When carried in a document (e.g., XRD or JRD) or protocol that supports the Unicode character set (e.g., UTF-8 or UTF-16 [21]), the URI strings may appear in the protocol or document's native encoding without percent-encoding. Such usage of a URI is commonly referred to as an Internationalized Resource Identifier (IRI). Conversion between an IRI and URI is described in Section 3 of RFC 3987 [13].

6. The "acct" Link Relation

Users of some services might have an acct URI that looks significantly different from their email address, perhaps using entirely different domain names. It may be useful to allow the mapping of an assumed account identifier to the correct account identifier.

Some users may also hold multiple different accounts and would like to allow users to find information distributed across multiple accounts.

To accomplish either of these two objectives, one uses the "acct" link relation. Consider the following example.

Suppose Alice receives an email from bob@example.net. While Bob's email identifier might be in the example.net domain, he holds his account with an acct URI in the example.com domain. His email provider may provide Webfinger services to enable redirecting Alice when she queries for acct:bob@example.net.

Suppose Alice issues the following request:

```
GET /.well-known/host-meta.json?resource=\
                        acct%3Abob%40example.net HTTP/1.1
Host: example.net
```

The response that Alice receives back might be:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=UTF-8

{
  "subject" : "acct:bob@example.net",
  "links" :
  [
    {
      "rel" : "acct",
      "href" : "acct:bob@example.com"
    }
```

```
    ]
  }
```

Alice's Webfinger client could then perform another query against the URI acct:bob@example.com in order to get the information she is seeking.

Webfinger clients need to take steps to avoid getting into loops where two accounts, directly or indirectly, refer the client to each other.

There are no limits on the number of acct link relations that might be returned in a Webfinger query.

7. Cross-Origin Resource Sharing (CORS)

Webfinger is most useful when it is accessible without restrictions on the Internet, and that includes web browsers. Therefore, Webfinger servers MUST support Cross-Origin Resource Sharing (CORS) [7]. Specifically, all queries to /.well-known/host-meta, /.well-known/host-meta.json, and to the LRDD URI must include the following HTTP header in the response:

```
Access-Control-Allow-Origin: *
```

QUESTION: Do we want to require CORS? Do we want to make it a SHOULD? Or, do we want to say nothing about CORS?

8. Security Considerations

All of the security considerations applicable to Web Host Metadata [9] and Cross-Origin Resource Sharing [7] are also applicable to this specification. Of particular importance is the recommended use of HTTPS to ensure that information is not modified during transit. Clients should verify that the certificate used on an HTTPS connection is valid.

When using HTTP to request an XRD document, Webfinger clients SHOULD verify the XRD document's signature, if present, to ensure that the XRD document has not been modified. Webfinger servers SHOULD include a signature for XRD documents.

Service providers and users should be aware that placing information on the Internet accessible through Webfinger means that any user can access that information. While Webfinger can be an extremely useful tool for allowing quick and easy access to one's avatar, blog, or other personal information, users should understand the risks, too. If one does not wish to share certain information with the world, do not allow that information to be accessible through Webfinger.

The easy access to user information via Webfinger was a design goal of the protocol, not a limitation. If one wishes to limit access to information available via Webfinger, such as a Webfinger server for use inside a corporate network, the network administrator must take measures necessary to limit access from outside the network.

9. IANA Considerations

RFC Editor: Please replace QQQQ in the following two sub-sections with a reference to this RFC.

9.1. Registration of the "acct" URI scheme name

This specification requests IANA to register the "acct" URI scheme in the "Permanent URI Schemes" sub-registry in the "Uniform Resource Identifier (URI) Schemes" IANA registry [17]. This registration follows the URI Scheme Registration Template detailed in Section 5.4 of RFC 4395 [15].

URI scheme name: acct

Status: Permanent

URI scheme syntax: see Section 4.1 of RFC QQQQ

URI scheme semantics: see Section 4.1 of RFC QQQQ

Encoding considerations: The "acct" URI scheme allows any character from the Unicode character set encoded as a UTF-8 string that is then percent-encoded as necessary to result in an internal representation in US-ASCII [10]

Applications/protocols that use this URI scheme name: Webfinger

Security considerations: see Section 7 of RFC QQQQ

Contact: Gonzalo Salgueiro <gsalguei@cisco.com>

Author/Change controller: IETF <ietf@ietf.org>

References: See Section 10 of RFC QQQQ

9.2. Registration of the "acct" Link Relation Type

Relation Name: acct

Description: A link relation that refers to a user's Webfinger account identifier.

Reference: RFC QQQQ

Notes:

Application Data:

10. Acknowledgments

The authors would like to acknowledge Eran Hammer-Lahav, Blaine Cook, Brad Fitzpatrick, and Laurent-Walter Goix for their invaluable input.

11. References

11.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[3] Nottingham, M., "Web Linking", RFC 5988, October 2010.

[4] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[5] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[6] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[7] Van Kesteren, A., "Cross-Origin Resource Sharing", W3C CORS http://www.w3.org/TR/cors/, July 2010.

[8] Hammer-Lahav, E. and W. Norris, "Extensible Resource Descriptor (XRD) Version 1.0", <http://docs.oasis-open.org/xri/xrd/v1.0/xrd-1.0.html>.

[9] Hammer-Lahav, E. and Cook, B., "Web Host Metadata", RFC 6415, October 2011.

[10] American National Standards Institute, "Coded Character Set - 7-bit American Standard Code for Information Interchange", ANSI X3.4, 1986.

[11] The Unicode Consortium. The Unicode Standard, Version 6.1.0, (Mountain View, CA: The Unicode Consortium, 2012. ISBN 978-1-936213-02-3) http://www.unicode.org/versions/Unicode6.1.0/.

[12] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.

[13] Duerst, M., "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

11.2. Informative References

[14] Zimmerman, D., "The Finger User Information Protocol", RFC 1288, December 1991.

[15] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 35, RFC 4395, February 2006.

[16] Perreault, S., "vCard Format Specification", RFC 6350, August 2011.

[17] Internet Assigned Numbers Authority (IANA) Registry, "Uniform Resource Identifier (URI) Schemes", <http://www.iana.org/assignments/uri-schemes.html>.

[18] "Transport Independent, Printer/System Interface", IEEE Std 1284.1-1997, 1997.

[19] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, November 2003.

[20] Information Systems -- Coded Character Sets 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII), ANSI X3.4-1986, December 30, 1986.

[21] Hoffman, P., Yergeau, F., "UTF-16, an encoding of ISO 10646", RFC 2781, February 2000.

Author's Addresses

Paul E. Jones
Cisco Systems, Inc.
7025 Kit Creek Rd.
Research Triangle Park, NC 27709
USA

Phone: +1 919 476 2048
Email: paulej@packetizer.com
IM: xmpp:paulej@packetizer.com

Gonzalo Salgueiro
Cisco Systems, Inc.
7025 Kit Creek Rd.
Research Triangle Park, NC 27709
USA

Phone: +1 919 392 3266
Email: gsalguei@cisco.com
IM: xmpp:gsalguei@cisco.com

Joseph Smarr
Google

Email: jsmarr@google.com