Network Working Group
Internet Draft
Intended status: Standards Track
Expires: April 22, 2013

Paul E. Jones
Gonzalo Salgueiro
Cisco Systems
Joseph Smarr
Google

October 22, 2012

WebFinger
draft-ietf-appsawg-webfinger-02.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 discover 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 April 22, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction....2
2. Terminology....3
3. Overview....3
4. Example Uses of WebFinger....4
4.1. Locating a User's Blog....4
4.2. Simplifying the Login Process....7
4.3. Retrieving Device Information....8
5. WebFinger Protocol....8
5.1. Performing a WebFinger Query....9
5.2. The Web Host Metadata "resource" Parameter....10
5.3. The Web Host Metadata "rel" Parameter....12
5.4. WebFinger and URIs....14
6. The "acct" Link Relation....14
6.1. Purpose for the "acct" Link Relation....14
6.2. Example Message Exchange Using the "acct" Link Relation..15
7. Cross-Origin Resource Sharing (CORS)....16
8. Controlling Access to Information....17
9. Hosted and Distributed WebFinger Services....17
9.1. Hosting the Entire Domain....17
9.2. Distributed WebFinger Services....18
10. Web Host Metadata Interoperability Considerations....20
11. Security Considerations....20
12. IANA Considerations....21
12.1. Registration of the "acct" Link Relation Type....21
13. Acknowledgments....21
14. References....21
14.1. Normative References....21
14.2. Informative References....22
APPENDIX A: XRD Usage (Non-normative)....24
A.1. How XRD Documents are Requested via WebFinger....24
A.2. WebFinger Example using XRDs....24
A.3. Security Considerations Related to XRDs....25
Author's Addresses....26

1. Introduction

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

.plan in the user's home directory.

WebFinger borrows the concept of the legacy finger protocol, but introduces a very different approach to sharing information. Rather than return a simple unstructured text file, Webfinger uses structured documents that contain link relations. These link relations point to information and might return properties related 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 discover 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 identity 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].

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 [4], the attribute is a "rel" and the value is an "href".

3. Overview

WebFinger enables the discovery of information about accounts, devices, and other entities that are associated with a host. Discover involves two distinct steps that may be optimized as a single step, as will be explained later. The first step is to query the host to find out how to discover information about accounts, devices, and other entities associated with that host. The second step is to query explicitly for a specific resource (e.g., user account) to discover a set of link relations that point to resource-specific information about the entity being queried.

This protocol makes heavy use of well-known URIs as defined in RFC 5785 [3] and "Link Relations" as defined in RFC 5988 [4]. Further, the protocol builds on RFC 6415 [11], which provides the foundation for the procedures described in this document.

Briefly, a link is a typed connection between two web resources that are identified by Internationalized Resource Identifiers (IRIs) [13]; this connection consists of a context IRI, a link relation type, a

target IRI, and optionally some target attributes, resulting in statements of the form "{context IRI} has a {relation type} resource at {target IRI}, which has {target attributes}". When used in the Link HTTP header, the context IRI is the IRI of the requested resource, the relation type is the value of the "rel" parameter, the target IRI is URI-Reference contained in the Link header, and the target attributes are the parameters such as "hreflang", "media", "title", "title*", "type", and any other link-extension parameters.

The framework for WebFinger consists of several building blocks:

1. To query the host, one requests a web host metadata document located at the well-known URI /.well-known/host-meta or /.well-known/host-meta.json (referred to as the host-meta resources) at the host.
2. The web server at the host returns a JavaScript Object Notation (JSON) [5] Resource Descriptor (JRD) or an Extensible Resource Descriptor (XRD) [10] document, including a Link-based Resource Descriptor Document (LRDD) link relation.
3. To discover information about accounts, devices, or other entities associated with the host, one requests the actual Link-based Resource Descriptor Document associated with a particular URI at the host (e.g., an "acct" URI, "http" URI, or "mailto" URI).
4. The web server at the host returns a JRD or XRD document for the requested URI, which includes link relations pointing to resources that contain more detailed information about the entity.

This model is illustrated in the examples in Section 4, then described more formally in Section 5. Steps 2 and 3 above can be accomplished simultaneously by utilizing the "resource" parameter defined in Section 5.2.

4. 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.

All of the following examples utilize JRDs, as that is the only mandatory format required to be supported by WebFinger servers. For completeness, an example utilizing XRDs is presented in Appendix A.

4.1. 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. After receiving the message from Bob (bob@example.com), your email client performs the following steps behind the scenes.

First, your email client tries to get the host metadata information for the host example.com. It does this by issuing the following HTTPS query to example.com:

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

The server replies with a JRD document:

```
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/?f=json&uri={uri}"
    }
  ]
}
```

The client then processes the received JRD in accordance with the Web Host Metadata procedures. The client will see the LRDD link relation and issue a query with the user's account URI [6] or other URI that serves as an alias for the account. (The account URI is discussed in Section 4.2.) The query might look like this:

```
GET /lrdd/?f=json&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/json; charset=UTF-8

{
  "expires" : "2012-10-12T20:56:11Z",
  "subject" : "acct:bob@example.com",
```

```
      "aliases" :
      [
        "http://www.example.com/~bob/"
      ],
      "links" :
      [
        {
          "rel" : "http://webfinger.net/rel/avatar",
          "href" : "http://www.example.com/~bob/bob.jpg"
        },
        {
          "rel" : "http://webfinger.net/rel/profile-page",
          "href" : "http://www.example.com/~bob/"
        },
        {
          "rel" : "http://packetizer.com/rel/blog",
          "href" : "http://blogs.example.com/bob/"
        },
        {
          "rel" : "vcard",
          "href" : "http://www.example.com/~bob/bob.vcf"
        }
      ]
    }
```

The email client might take note of the "blog" link relation in the above JRD 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. Lastly, the client might consider the vcard [16] link relation in order to update contact information for Bob.

Note in the above example that an alias is provided that can also be used to return information about the user's account. Had the "http:" URI shown as an alias been used to query for information about Bob, the query would have appeared as:

```
  GET /lrdd/?uri=http%3A%2F%2Fwww.example.com%2F~bob%2F HTTP/1.1
  Host: example.com
```

The response would have been substantially the same, with the subject and alias information changed as necessary. Other information, such as the expiration time might also change, but the set of link relations and properties would be the same with either response.

4.2. 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 discover the OpenID identity provider URL for Carol and to get Carol's avatar. In this example, we utilize the "rel" and "resource" parameters as described in sections 5.2 and 5.3:

```
GET /.well-known/host-meta.json?\
    rel=avatar%20\
    http%3A%3F%3Fspecs.openid.net%3Fauth%3F2.0%3Fprovider&\
    resource=acct%3Acarol%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: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. Her avatar can also be displayed for the benefit of other users on the blog.

4.3. 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. Any URI scheme that contains host information MAY be used with WebFinger. 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 like device:p1.example.com. While the "device" URI scheme is not presently specified, we use it here only for illustrative purposes.

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

```
GET /.well-known/host-meta.json?resource=\
                  device%3Ap1.example.com HTTP/1.1
Host: example.com
```

The link relations that are returned may be quite different than those for user accounts. 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.

5. WebFinger Protocol

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

While WebFinger strives to maintain backward-compatibility with RFC 6415, this specification introduces a fundamental change in requirements. Specifically, support for server-side production of JSON Resource Descriptor (JRD) documents is mandatory and support for server-side production Extensible Resource Descriptor (XRD) documents is optional. Please refer to Section 10 for interoperability considerations.

5.1. Performing a WebFinger Query

The first step a client performs 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 specification. It is strongly RECOMMENDED that WebFinger servers return content using secure (HTTPS) connections. Clients MUST first attempt queries using HTTPS before attempting a query using HTTP.

WebFinger clients MUST locate the LRDD link relation 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 MAY accept requests for both JRD and XRD documents, but MUST support requests for JRD documents. For interoperability with RFC 6415 implementations, the default representation returned by a server via the resource at /.well-known/host-meta MUST be an XRD document if XRD is supported by the server and a JRD document is not explicitly requested by the client. The default format returned via the resource /.well-known/host-meta.json MUST be a JRD document.

As per RFC 6415, a JRD document MUST be returned by the WebFinger server if the client explicitly requests it by querying /.well-known/host-meta.json or by querying /.well-known/host-meta and including an "Accept" header in the HTTP request with a type of "application/json" [5]. Additionally, the server MUST return a JRD document if it does not support production of XRD documents (or any other format requested by the client). Servers MUST indicate the type of document returned using the "Content-Type" header in the HTTP response.

To avoid the possibility of receiving the wrong document format, WebFinger clients SHOULD submit queries to the server via the /.well-known/host-meta.json resource.

If the client requests a JRD document when querying for host metadata, the WebFinger server MUST assume that the client will want a JRD document when querying the LRDD resource. Thus when the WebFinger server returns a JRD document containing host metadata that

contains an LRDD link relation, it MUST include a URI for the LRDD

resource(s) that will return a JRD document. Likewise, if a client requests an XRD document when querying the host metadata resource, the server MUST, unless unable due to external factors, return LRDD link relations that would return XRD documents.

It is important to note that unless the "resource" parameter is used as per section 5.2, it is the responsibility of the client to process each of the LRDD link relations as per Section 4.2 of RFC 6415 if a server returns multiple LRDD link relations. Multiple LRDD link relations in a server response do not represent alternative URIs for the same LRDD 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 MUST result in the server returning a 404 status code.

WebFinger servers MAY include cache validators in a response to enable conditional requests by clients and/or expiration times as per RFC 2616 section 13.

5.2. The Web Host Metadata "resource" Parameter

In addition to the traditional 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 response. This parameter essentially pushes the work to the server to form a complete resource descriptor for the specified resource.

WebFinger servers compliant with this specification MUST support for the "resource" parameter. Note that an RFC 6415-compliant server might not implement the "resource" parameter, though the server would respond to queries from the client as described in RFC 6415. Thus, WebFinger clients MUST check the server response to ensure that the "resource" parameter is supported as explained below.

To utilize the host-meta "resource" parameter, a WebFinger client issues a request to /.well-known/host-meta.json (RECOMMENDED) or /.well-known/host-meta 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
```

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; and

* Collect and expand all resource-specific link relations, including those returned by querying for any LRDD link relations, discard any host-wide link relations, and return a complete resource descriptor following the processing rules in Section 4.2 of RFC 6415; and

It is not the responsibility of the WebFinger server to verify, for example, that a URI pointing to a person's avatar is a valid URI. If the WebFinger server needs to query an LRDD resource to collect additional resource-specific information, any errors (e.g., 500 or 404) MUST be ignored by the server. When a request for an LRDD fails, the server MUST NOT attempt to augment missing resource information or return a "template" type link relation to a client that utilizes the "resource" parameter. Note that a WebFinger server might be implemented such that all LRDD resource-specific information can be resolved without issuing HTTP requests. How a WebFinger server collects and expands such resource-specific link relations is an implementation matter.

The WebFinger client MUST 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. The Subject would be absent if the "resource" parameter is not supported.

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
```

Note: The "\" character shown above and used throughout this document indicates that the line breaks at this point and continues on the next line. The content of the next line should be concatenated to the previous line without any whitespace characters, replacing the "\" character. This is shown only to avoid line wrapping in this document.

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",
      "href" : "http://example.com/author?\
                          q=http%3A%2F%2Fexample.com%2Fxy"
    }
  ]
}
```

5.3. The Web Host Metadata "rel" Parameter

WebFinger also defines the "rel" parameter for use when querying for host metadata or resource-specific information. It is used to return a subset of the information that would otherwise be returned without the "rel" parameter. When the "rel" parameter is used, only the link relations that match the space-separated list of link relations provided via "rel" are included in the list of links returned in the resource descriptor. All other information normally present in a resource descriptor is present in the resource descriptor, even when "rel" is employed.

The purpose of the "rel" parameter is to return a subset of resource's link relations. It is not intended to reduce the work required of a server to produce a response. That said, use of the parameter might reduce processing requirements on either the client

or server, and it might also reduce the bandwidth required to convey

the partial resource descriptor, especially if there are numerous link relation values to convey for a given resource.

Support for the "rel" parameter is OPTIONAL, but support is RECOMMENDED for the host-meta resources and LRDD resources.

For illustrative purposes, the following is an example usage of the "rel" parameter that aligns with the example in Section 1.1.1 of RFC 6415. The WebFinger client would issue this request to receive links that are of the type "hub" and "copyright":

```
GET /.well-known/host-meta.json?resource=\
      http%3A%2F%2Fexample.com%2Fxy&rel=hub%20copyright 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"
    }
  ]
}
```

Note that in this example, the "author" links are removed, though all other content is present. Since there were no "copyright" links, none are returned.

In the event that a client requests links for link relations that are not defined for the specified resource, a resource descriptor MUST be returned, void of any links. When a JRD is returned, the "links" array MAY be either absent or empty. The server MUST NOT return a 404 status code when a particular link relation specified via "rel"

is not defined for the resource, as a 404 status code is reserved for indicating that the resource itself (e.g., either /.well-known/host-meta.json or the resource indicated via the "resource" parameter) does not exist.

5.4. WebFinger and URIs

Requests for both LRDD documents and host metadata can include a parameter specifying the URI of an account, device, or other entity (for LRDD this is the "uri" parameter as defined by the operative JRD or XRD template and for host metadata this is the "resource" parameter). WebFinger itself is agnostic regarding the scheme of such a URI: it could be an "acct" URI [7], an "http" or "https" URI, a "mailto" URI, or some other scheme.

For resources associated with a user account at a host, use of the "acct" URI scheme is RECOMMENDED, since it explicitly identifies an account accessible via WebFinger. Further, the "acct" URI scheme is not associated with other protocols as, by way of example, the "mailto" URI scheme is associated with email. Since not every host offers email service, using the "mailto" URI scheme [8] is not ideal for identifying user accounts on all hosts. That said, use of the "mailto" URI scheme would be ideal for use with WebFinger to discover mail server configuration information for a user, for example.

A host MAY utilize one or more URIs that serve as aliases for the user's account, such as URIs that use the "http" URI scheme [2]. A WebFinger server MUST return substantially the same response to both an "acct" URI and any alias URI for the account, including the same set of link relations and properties. In addition, the server SHOULD include the entire list aliases for the user's account in the JRD or XRD returned when querying the LRDD resource or when utilizing the "resource" parameter.

6. The "acct" Link Relation

6.1. Purpose for the "acct" Link Relation

Users of some services might have an "acct" URI that looks significantly different from his or her email address, perhaps using an entirely different domain name. It is also possible for a user to have multiple accounts that a user wants to have cross-referenced from another account. To address both of these needs, this specification defines the "acct" link relation.

The "acct" link relation allows a resource descriptor to reference one or more other user account URIs. The "acct" link relation is intended to allow a client to incorporate additional link relations by reference so that it might utilize a more complete set of link

relations for a user. For example, a user acct:bob@example.com might wish to allow a client to discover additional information about him by including an "acct" link relation with the URI acct:bob@example.net.

Note that the "acct" link relation does not replace the use of standard HTTP 3xx response codes to indicate the new temporary or permanent location of a user account. If a user account is moved to a different location, then a 3xx response code SHOULD be used. Also, the "acct" link relation does not replace Link-based Resource Descriptor Documents (LRDDs). A WebFinger server might return multiple LRDD link relations for a user, each of which perhaps containing link relations that are to be merged to form a complete resource descriptor. The "acct" link relation is different in that it would refer to an entirely different, separate resource descriptor. Further, only a client would act consider the "acct" link relations as it performs queries, not the WebFinger server.

Since an account may make a reference to one or more different accounts, WebFinger clients that support automatic processing of the "acct" link relations MUST take steps to avoid loops wherein two account URIs, 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.

An "acct" link relation used within the context of a WebFinger query for a user's account MUST NOT return "acct" link relations for another user.

Client-side consideration of the "acct" link relation is OPTIONAL and WebFinger server MUST NOT assume a client will perform additional processing in response to receiving an "acct" link relation.

6.2. Example Message Exchange Using the "acct" Link Relation

Consider the following non-normative example.

Suppose Alice receives an email from bob@example.net. While Bob's email identifier might be in the example.net domain, he holds a user account in the example.com domain and another account in the example.org domain. His email provider may provide WebFinger services, but is unable to serve information from other domains.

Suppose Alice's client 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's client 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"
    },
    {
      "rel" : "acct",
      "href" : "acct:bob@example.org"
    },
    {
      "rel" : "acct",
      "href" : "mailto:bob@example.net"
    }
  ]
}
```

While these link relations provide Alice with very little information, Alice's WebFinger client could then perform subsequent queries against the URIs acct:bob@example.com, acct:bob@example.org, and mailto:bob@example.net in order to get the information Alice is seeking.

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) [9] when serving content intended for public consumption. Specifically, all queries to /.well-known/host-meta.json, /.well-known/host-meta, and to any LRDD URIs MUST include the following HTTP header in the response:

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

Enterprise WebFinger servers that wish to restrict access to information from external entities SHOULD use a more restrictive Access-Control-Allow-Origin header and MAY exclude the header entirely.

8. Controlling Access to Information

As with all web resources, access to the Host Metadata resource and the LRDD resource MAY require authentication. Further, failure to provide required credentials MAY result in the server forbidding access or providing a different response than had the client authenticated with the server.

Likewise, a server MAY provide different responses to different clients based on other factors, such as whether the client is inside or outside a corporate network. As a concrete example, a query performed on the internal corporate network might return link relations to employee pictures whereas link relations for employee pictures might not be provided to external entities.

Further, link relations provided in a WebFinger server response MAY point to web resources that impose access restrictions. For example, it is possible that the aforementioned corporate server may provide both internal and external entities with URIs to employee pictures, but further authentication MAY be required in order for the WebFinger client to access those picture resources if the request comes from outside the corporate network.

The decisions made with respect to what set of link relations a WebFinger server provides to one client versus another and what resources require further authentication, as well as the specific authentication mechanisms employed, are outside the scope of this document.

9. Hosted and Distributed WebFinger Services

9.1. Hosting the Entire Domain

As with most services provided on the Internet, it is possible for a domain owner to utilize "hosted" WebFinger services. By way of example, a domain owner might control most aspects of their domain, but use a third-party hosting service email. In the case of email, mail servers for a domain are identified by MX records. An MX record points to the mail server to which mail for the domain should be delivered. It does not matter to the sending mail server whether those MX records point to a server in the destination domain or a different domain.

Likewise, a domain owner might utilize the services of a third party to provide WebFinger services on behalf of its users. Just as a domain owner was required to insert MX records into DNS to allow for hosted email serves, the domain owner is required to redirect HTTP(S) queries to its domain to allow for hosted WebFinger services.

When a query is issued to /.well-known/host-meta.json or /.well-known/host-meta, the target domain's web server MUST return a 301, 302, or 307 response status code that includes a Location header pointing to the location of the hosted WebFinger service URL. The WebFinger service URL does not need to point to /.well-known/* on the hosting service provider server. In fact, it should not, as that location would be reserved for queries relating to the service provider's domain. WebFinger clients MUST follow all 301, 302, or 307 redirection requests.

As an example, let's assume that example.com's WebFinger services are hosted by example.net. Suppose a client issues a query for acct:alice@example.com like this:

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

The server might respond with this:

```
HTTP/1.1 301 Moved Permanently
Location: http://wf.example.net/example.org/host-meta.json
```

The client should follow the request, re-issuing the request to the URL provided in the Location header.

Note that both of the /.well-known/host-meta.json and /.well-known/host-meta resources need to be considered when redirecting request to third party service providers. Those URLs requests SHOULD NOT be redirected to the same location and without any differentiation, since the default format returned by host-meta.json is a JRD and the default format returned by host-meta MAY be XRD. Each resource is distinct and should be redirected separately and to different service locations or differentiated with a URI parameter. Since the "Referer" HTTP header field is not mandatory, service providers cannot rely on that header to determine the URL of the original request.

9.2. Distributed WebFinger Services

A domain owner may wish to manage only a part of its WebFinger services and WebFinger service providers or the domain owner may wish to distribute WebFinger services across a number of WebFinger service locations. The key to enabling this type of distribution is placement of resource-specific information in more than one LRDD document, each document existing at different locations.

Assume that the company operating example.com manages its own WebFinger services, but also wants to utilize the services of

example.org to serve link relations related to some aspects of its business. Suppose a client issued this request:

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

The server might reply with this JRD document:

```
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/?f=json&uri={uri}"
    },
    {
      "rel" : "lrdd",
      "type" : "application/json",
      "template" : "https://wf.example.org/lrdd/?f=json&uri={uri}"
    }
  ]
}
```

This would indicate to the client that some of the resource-specific information is found at example.com and some is found at example.org, following those specific URLs. Observing the rules in Section 4.2 of RFC 6415, the client would issue queries to both URLs and construct a complete resource descriptor.

As discussed in Section 5.2, a client may issue a query like this to the example.com domain:

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

In that case, it would be the responsibility of the WebFinger server at example.com to query the LRDD URL at example.org and then compose a complete descriptor document. The client that uses the resource parameter remains entirely oblivious to the fact that link relation information is distributed across multiple servers or domains.

10. Web Host Metadata Interoperability Considerations

As noted in Section 3, RFC 6415 required all servers to support the production of Extensible Resource Documents (XRDs) and optionally support the production of JSON Resource Documents (JRDs). This specification reverses that requirement: WebFinger-compliant servers MUST support JRD and MAY support XRD documents.

Given that some servers might implement only RFC 6415 and other servers might implement only the minimum required set of features defined for WebFinger, all clients should take care to ensure to request a resource descriptor in the appropriate format. If a client wishes to receive only JRDs, for example, it SHOULD issue a request to /.well-known/host-meta.json, but MAY issue a request to /.well-known/host-meta and include the "Accept" header with the type "application/json".

Further, clients MUST ensure that the response returned from the server contains the correct format. RFC 6415-compliant servers might return an XRD document, regardless of what is requested by the client.

Lastly, RFC 6415 did not require clients to follow 301, 302, or 307 redirection requests, but WebFinger clients MUST re-issue requests when redirected using any of those HTTP status codes.

11. Security Considerations

All of the security considerations applicable to Web Host Metadata and Cross-Origin Resource Sharing [9] 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.

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 freely accessible through WebFinger.

The aforementioned word of caution is perhaps worth emphasizing again with respect to dynamic information one might wish to share, such as the current location of a user. WebFinger can be a powerful tool used to assemble information about a person all in one place, but service providers and users should be mindful of the nature of that information shared and the fact that it might be available for the

entire world to see. Sharing location information, for example, would potentially put a person in danger from any individual who might seek to inflict harm on that person.

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. Using standard methods for securing web resources, network administrators do have the ability to control access to resources that might return sensitive information. Further, WebFinger servers can be employed in such a way as to require authentication and prevent disclosure of information to unauthorized entities.

12. IANA Considerations

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

12.1. 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:

13. Acknowledgments

The authors would like to acknowledge Eran Hammer-Lahav, Blaine Cook, Brad Fitzpatrick, Laurent-Walter Goix, Joe Clarke, Mike Jones, and Peter Saint-Andre for their invaluable input.

14. References

14.1. 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., Hammer-Lahav, E., "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.

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

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

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

[7] Saint-Andre, P., "The 'acct' URI Scheme", draft-ietf-appsawg-acct-uri-00, August 2012.

[8] Duerst, M., Masinter, L., and J. Zawinski, "The 'mailto' URI Scheme", RFC 6068, October 2010.

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

[10] 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.

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

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

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

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

APPENDIX A: XRD Usage (Non-normative)

A.1. How XRD Documents are Requested via WebFinger

The framework for using XRD documents with WebFinger is as follows:

1. WebFinger clients issue request for XRD documents by requesting the Web Host Metadata document located at the well-known URI /.well-known/host-meta at the host.
2. The web server at the host returns an XRD document, including a Link-based Resource Descriptor Document (LRDD) link relation.
3. To discover information about accounts, devices, or other entities associated with the host, a request is issued for the Link-based Resource Descriptor Document(s) associated with a particular URI at the host (e.g., an "acct" URI, "http" URI, or "mailto" URI).
4. The web server at the host would return an XRD document about the requested URI, which included those resource-specific link relations pointing to resources that contain information about the entity.
5. Following the procedures in Section 4.2 of RFC 6415, the client would assemble all of the resource-specific link relations from the host-meta resource and LRDD resource(s) into a complete resource descriptor.

The LRDD resources return resource descriptor documents of the type "application/xrd+xml".

A.2. WebFinger Example using XRDs

Section 4 introduces examples where JRD documents are returned to clients. For completeness, this section shows an example where a client requests an XRD document.

Recall the example from Section 4.1 where the email client tried to retrieve information about Bob to discover the URL for his blog. If the client implemented support for XRD, it tries to get the host metadata information for the domain example.com in a similar way. As with the original example, it issues the following HTTPS query to example.com:

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

The server replies with an XRD 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 procedures. The client will see the LRDD link relation and issue a query with the user's account URI [6] or other URI that serves as an alias for the account. (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">
  <Expires>2012-10-12T20:56:11Z</Expires>
  <Subject>acct:bob@example.com</Subject>
  <Alias>http://www.example.com/~bob/</Alias>
  <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="http://packetizer.com/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.

A.3. Security Considerations Related to XRDs

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. Additionally, WebFinger servers SHOULD include a signature for XRD documents served over HTTP.

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