Network Working Group
Internet Draft
Intended status: Standards Track
Expires: May 29, 2013

Paul E. Jones
Gonzalo Salgueiro
Cisco Systems
Joseph Smarr
Google

November 29, 2012

WebFinger
draft-ietf-appsawg-webfinger-06.txt

Abstract

This specification defines the WebFinger protocol, which can be used to discover information about people or other entities on the Internet using standard HTTP methods.

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 May 29, 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....2
3. Overview....3
4. Example Use of WebFinger....3
4.1. Locating a User's Blog....3
4.2. Identity Provider Discovery for OpenID Connect....5
4.3. Auto-Configuration of Email Clients....6
4.4. Retrieving Device Information....7
5. WebFinger Protocol....8
5.1. Performing a WebFinger Query....8
5.2. The JSON Resource Descriptor (JRD) Document....9
5.3. The "rel" Parameter....9
5.4. WebFinger and URIs....11
6. Cross-Origin Resource Sharing (CORS)....12
7. Access Control....12
8. Hosted WebFinger Services....13
9. Security Considerations....13
10. IANA Considerations....15
11. Acknowledgments....15
12. References....15
12.1. Normative References....15
12.2. Informative References....16
Author's Addresses....17

1. Introduction

WebFinger is used to discover information about people or other entities on the Internet using standard HTTP [2] methods. The WebFinger server returns a structured document that contains link relations, properties, and/or other information that is suitable for automated processing. For a person, the kinds of information that might be shared via WebFinger include a personal profile address, identity service, telephone number, or preferred avatar. For other entities on the Internet, the server might return documents containing link relations that allow a client to discover 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., looking up someone'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) and Web Linking [4], the attribute is a "rel" and the value is an "href". WebFinger also uses the "rel" attribute, where the "rel" value is either a single IANA-registered link relation type [12] or a URI [6].

3. Overview

WebFinger enables the discovery of information about users, devices, and other entities that are associated with a host. Discovery involves a single HTTP GET request to the well-known [3] "webfinger" resource at the target host and receiving back a JavaScript Object Notation (JSON) [5] Resource Descriptor (JRD) document [11] containing link relations. The request MUST include the URI or IRI [7] for the entity for which information is sought as a parameter named "resource".

Use of WebFinger is illustrated in the examples in Section 4, then described more formally in Section 5.

4. Example Use of WebFinger

This non-normative section shows a few sample uses of WebFinger.

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.

Assume your email client can discover the blog for you. After receiving the message from Bob (bob@example.com), you instruct your email client to perform a WebFinger query. It does so by issuing the following HTTPS query to example.com:

```
GET /.well-known/webfinger?
                      resource=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" : "2012-11-16T19:41:35Z",
      "subject" : "acct:bob@example.com",
      "aliases" :
      [
        "http://www.example.com/~bob/"
      ],
      "properties" :
      {
          "http://example.com/rel/role/" : "employee"
      },
      "links" :
      [
        {
          "rel" : "http://webfinger.net/rel/avatar",
          "type" : "image/jpeg",
          "href" : "http://www.example.com/~bob/bob.jpg"
        },
        {
          "rel" : "http://webfinger.net/rel/profile-page",
          "href" : "http://www.example.com/~bob/"
        },
        {
          "rel" : "blog",
          "type" : "text/html",
          "href" : "http://blogs.example.com/bob/",
          "titles" :
          {
              "en-us" : "The Magical World of Bob",
              "fr" : "Le monde magique de Bob"
          }
        },
        {
          "rel" : "vcard",
          "href" : "http://www.example.com/~bob/bob.vcf"
        }
      ]
    }
```

The email client would 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 [14] link relation in order to update contact information for Bob.

In the above example, an "acct" URI [8] is used in the query, though any valid alias for the user might also be used. An alias is a URI

that is different from the "subject" URI that identifies the same

entity. In the above example, there is one "http" alias returned, though there might have been more than one. Had the "http:" URI shown as an alias been used to query for information about Bob, the query would have appeared as:

```
GET /.well-known/webfinger?
         resource=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. Identity Provider Discovery for OpenID Connect

Suppose Carol wishes to authenticate with a web site she visits using OpenID Connect [16]. She would provide the web site with her OpenID Connect identifier, say carol@example.com. The visited web site would perform a WebFinger query looking for the OpenID Connect Provider. Since the site is interested in only one particular link relation, the server might utilize the "rel" parameter as described in section 5.3:

```
GET /.well-known/webfinger?
  resource=acct%3Acarol%40example.com&
  rel=http%3A%2F%2Fopenid.net%2Fspecs%2Fconnect%2F1.0%2Fissuer
  HTTP/1.1
Host: example.com
```

The server might respond with a JSON Resource Descriptor document like this:

```
{
  "expires" : "2012-11-16T19:41:35Z",
  "subject" : "acct:carol@example.com",
  "aliases" :
  [
    "http://www.example.com/~carol/"
  ],
  "properties" :
  {
      "http://example.com/rel/role/" : "employee"
  },
  "links" :
  [
    {
      "rel" : "http://openid.net/specs/connect/1.0/issuer",
```

```
           "href" : "https://openid.example.com/"
```

```
      }
    ]
  }
```

Since the "rel" parameter only filters the link relations returned by the server, other name/value pairs in the response, including any aliases or properties, would be returned. Also, since support for the "rel" parameter is optional, the client must not assume the "links" array will contain only the requested link relation.

4.3. Auto-Configuration of Email Clients

WebFinger could be used to auto-provision an email client with basic configuration data. Suppose that sue@example.com wants to configure her email client. Her email client might issue the following query:

```
GET /.well-known/webfinger?
         resource=mailto%3Asue%40example.com HTTP/1.1
Host: example.com
```

The response from the server would contain entries for the various protocols, transport options, and security options. If there are multiple options, the server might return a link relation that for each of the valid options and the client or Sue might select which option to choose. Since JRD documents list link relations in a specific order, then the most-preferred choices could be presented first. Consider this response:

```
{
  "subject" : "mailto:sue@example.com",
  "links" :
  [
    {
      "rel" : "http://example.net/rel/smtp-server",
      "properties" :
      {
        "host" : "smtp.example.com",
        "port" : "587",
        "login-required" : "yes",
        "transport" : "starttls"
      }
    },
    {
      "rel" : "http://example.net/rel/imap-server",
      "properties" :
      {
        "host" : "imap.example.com",
        "port" : "993",
```

```
           "transport" : "ssl"
```

```
      }
    }
  ]
}
```

In this example, you can see that the WebFinger server advertises an SMTP service and an IMAP service. In this example, the "href" entries associated with the link relation are absent. This is valid when there is no external reference that needs to be made.

4.4. Retrieving Device Information

As another example, suppose there are printers on the network and you would like to check the current toner level for a particular printer identified via the URI device:p1.example.com. While the "device" URI scheme is not presently specified, we use it here 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/webfinger?resource=
                  device%3Ap1.example.com HTTP/1.1
Host: example.com
```

The link relations that are returned for a device 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" : "http://example.com/rel/tipsi",
      "href" : "http://192.168.1.5/npap/"
    }
  ]
}
```

While this example is fictitious, you can imagine that perhaps the Transport Independent, Printer/System Interface [15] 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 is a simple HTTP-based web service that utilizes the JSON Resource Descriptor (JRD) document format and the Cross-Origin Resource Sharing (CORS) [10] specification.

This specification defines URI parameters that are passed from the client to the server when issuing a request. These parameters, "resource" and "rel", and the parameter values are included in the "query" component of the URI (see Section 3.4 of RFC 3986). To construct the "query" component, the client performs the following steps. First, each parameter value is percent-encoded as per Section 2.1 of RFC 3986. Next, the client constructs a string to be placed in the query component by concatenating the name of the first parameter together with an equal sign ("=") and the percent-encoded parameter value. For any subsequent parameters, the client appends an ampersand ("&") to the string, the name of the next parameter, an equal sign, and percent-encoded parameter value. The client MUST NOT insert any spaces while constructing the string. The order in which the client places each parameter and its corresponding parameter value is unspecified.

5.1. Performing a WebFinger Query

WebFinger clients issue queries to the well-known resource /.well-known/webfinger. All queries MUST include the "resource" parameter exactly once and set to the value of the URI for which information is being sought. If the "resource" parameter is absent or malformed, the WebFinger server MUST return a 400 status code.

Clients MUST first attempt a query the server using HTTPS and utilize HTTP only if an HTTPS connection cannot be established. If the HTTPS server has an invalid certificate or returns an HTTP status code indicating some error, including a 4xx or 5xx, the client MUST NOT use HTTP in attempt to complete the discovery.

WebFinger servers MUST return JRD documents as the default representation for the resource. A client MAY include the "Accept" header to indicate a desired representation, though no other representation is defined in this specification. For the JRD document, the media type is "application/json" [5].

If the client queries the WebFinger server and provides a URI for which the server has no information, the server MUST return a 404 status code.

WebFinger servers can 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 JSON Resource Descriptor (JRD) Document

The JSON Resource Descriptor (JRD) document is formally described in Appendix A of [11]. There is a RECOMMENDED order of JRD name/value pairs. Further, WebFinger requires some name/value pairs and some are optional. The following list indicates the preferred order and comments on the presence or absence of name/value pairs:

- "expires" (name/value pair) is optional
- "subject" (name/value pair) is required and MUST be the value of the "resource" parameter
- "aliases" (array) is optional and absence or an empty array are semantically the same
- "properties" (object) is optional and absence or an empty object are semantically the same
- "links" (array) is optional and absence or an empty array are semantically the same

Values within the "links" array are presented by the server in order of preference.

The "links" array is comprised of several name/value pairs. As above, the following list indicates the preferred order within a "links" array and comments on the presence or absence of name/value pairs within the array:

- "rel" (name/value pair) is required
- "type" (name/value pair) is optional
- "href" (name/value pair) is optional
- "template" (name/value pair) is forbidden
- "titles" (object) is optional and absence or an empty object are semantically the same
- "properties" (object) is optional and absence or an empty object are semantically the same

Clients MUST ignore any unknown or forbidden name/value pair received in the JRD document.

5.3. The "rel" Parameter

WebFinger defines the "rel" parameter to request only 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 link relations provided via "rel" are included in the array of links returned in the JSON Resource Descriptor document. All other information normally present in a resource descriptor is present in the resource descriptor, even when "rel" is employed.

The "rel" parameter MAY be transmitted to the server multiple times in order to request multiple types of link relations.

The purpose of the "rel" parameter is to return a subset of resource's link relations. 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 RECOMMENDED on the server. Should the server not support the "rel" parameter, it MUST ignore it and process the request as if any "rel" parameter values were not present.

The following example presents the same example as found in section 4.1, but uses the "rel" parameter in order to select two link relations:

```
GET /.well-known/webfinger?
           resource=acct%3Abob%40example.com&
           rel=http%3A%2F%2Fwebfinger.net%2Frel%2Fprofile-page&
           rel=vcard HTTP/1.1
Host: example.com
```

In this example, the client requests the link relations of type "http://webfinger.net/rel/profile-page" and "vcard". The server then responds with a message like this:

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

{
  "expires" : "2012-11-16T19:41:35Z",
  "subject" : "acct:bob@example.com",
  "aliases" :
  [
    "http://www.example.com/~bob/"
  ],
  "properties" :
  {
      "http://example.com/rel/role/" : "employee"
  },
  "links" :
  [
    {
      "rel" : "http://webfinger.net/rel/profile-page",
```

```
           "href" : "http://www.example.com/~bob/"
```

```
      },
      {
        "rel" : "vcard",
        "href" : "http://www.example.com/~bob/bob.vcf"
      }
    ]
  }
```

As you can see, the server returned only the link relations requested by the client, but also included the other parts of the JSON resource Descriptor document.

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. In the returned JRD, the "links" array MAY be absent, empty, or contain only links that did match a provided "rel" value. 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/webfinger or the resource indicated via the "resource" parameter) does not exist.

5.4. WebFinger and URIs

WebFinger requests can include a parameter specifying the URI of an account, device, or other entity. WebFinger 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 [9] 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.

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. The only name/value pairs in the response that MAY be different include "subject", "expires", and "aliases". In addition, the server SHOULD include the entire list aliases for the user's account in the JRD returned when querying the LRDD resource or when utilizing the "resource" parameter.

6. Cross-Origin Resource Sharing (CORS)

WebFinger resources might not be accessible from a web browser due to "Same-Origin" policies. The current best practice is to make resources available to browsers through Cross-Origin Resource Sharing (CORS) [10], and servers MUST include the Access-Control-Allow-Origin HTTP header in responses. Servers SHOULD support the least restrictive setting by allowing any domain access to the WebFinger resources:

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

There are cases where defaulting to the least restrictive setting is not appropriate, for example a WebFinger server on an intranet that provides sensitive company information should not allow CORS requests from any domain, as that could allow leaking of that sensitive information. WebFinger servers that wish to restrict access to information from external entities SHOULD use a more restrictive Access-Control-Allow-Origin header.

7. Access Control

As with all web resources, access to the /.well-known/webfinger 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, the aforementioned corporate server may provide both internal and external entities with URIs to employee pictures, but further authentication might be required in order for the client to access the 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.

8. Hosted WebFinger Services

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 for 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/webfinger, the 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. WebFinger clients MUST follow all 301, 302, or 307 redirection requests.

As an example, 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/webfinger?
              resource=acct%3Aalice%40example.com HTTP/1.1
Host: example.com
```

The server might respond with this:

```
HTTP/1.1 307 Temporary Redirect
Location: http://wf.example.net/example.com/webfinger?
                  resource=acct%3Aalice%40example.com HTTP/1.1
```

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

9. Security Considerations

Since this specification utilizes Cross-Origin Resource Sharing (CORS) [10], all of the security considerations applicable CORS are also applicable to this specification.

The recommended use of HTTPS is to ensure that information is not modified during transit. It should be appreciated that in environments where an HTTPS server is normally available, there exists the possibility that a compromised network might have its WebFinger server operating on HTTPS replaced with one operating only over HTTP. As such, clients that need to ensure data is not compromised SHOULD NOT issue queries over a non-secure connection. While Section 5.1 allows for clients that fail to establish an HTTPS connection to attempt a query using HTTP, a client and any underlying client libraries are not required to re-issue queries using HTTP and SHOULD NOT when security for a given application that uses WebFinger is paramount.

When using HTTPS, clients MUST 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 and do not use any service supporting WebFinger. Further, WebFinger servers MUST NOT be used to provide any personal information to any party unless explicitly or implicitly authorized by the person whose information is being shared. Implicit authorization can be determined by the user's voluntary utilization of a service as defined by that service's relevant terms of use or published privacy policy.

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.

Finally, a WebFinger server has no means of ensuring that information provided by a user is accurate. Likewise, neither the server nor the client can be absolutely guaranteed that information has not been manipulated either at the server or along the communication path between the client and server. Use of HTTPS helps to address some concerns with manipulation of information along the communication path, but it clearly cannot address issues where the server provided incorrect information, either due to being provided false information or due to malicious behavior on the part of the server administrator. As with any information service available on the Internet, users should wary of information received from untrusted sources.

10. IANA Considerations

This specification registers the "webfinger" well-known URI in the Well-Known URI Registry as defined by [3].

URI suffix: webfinger

Change controller: IETF

Specification document(s): RFC QQQ

Related information: The JSON Resource Descriptor (JRD) documents obtained via the WebFinger web service are described in RFC 6415 Appendix A and RFC QQQ.

[RFC EDITOR: Please replace "QQQ" references in this section with the number for this RFC.]

11. Acknowledgments

The authors would like to acknowledge Eran Hammer-Lahav, Blaine Cook, Brad Fitzpatrick, Laurent-Walter Goix, Joe Clarke, Michael B. Jones, Peter Saint-Andre, Dick Hardt, Tim Bray, and Joe Gregorio for their invaluable input.

12. References

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

[8] Saint-Andre, P., "The 'acct' URI Scheme", draft-ietf-appsawg-acct-uri-01, October 2012.

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

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

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

12.2. Informative References

[12] IANA, "Link Relations", http://www.iana.org/assignments/link-relations/.

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

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

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

[16] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, "OpenID Connect Messages 1.0", June 2012, http://openid.net/specs/openid-connect-messages-1_0.html.

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