Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 6, 2020

D. Fuelling
Ripple
May 05, 2020

PayID Discovery
draft-fuelling-payid-discovery-01

Abstract

This specification defines the PayID Discovery protocol, which can be used to discover information about a 'payid' URI using standard HTTP methods.

The primary use-case of this protocol is to define how to transform a PayID URI into a URL that can be used with other protocols.

Feedback

This specification is a part of the PayID Protocol [1] work. Feedback related to this specification should be sent to payid@ripple.com [2].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

PayID Discovery is used to transform a PayID URI [PAYID-URI] into a URL (defined below as a PayID Discovery URL) that can then be used by higher-order protocols to discover metadata about a PayID-enabled service provider.

This document specifies two modes of PayID discovery: one using Webfinger [RFC7033] to resolve a corresponding PayID Discovery URL from a PayID using an interactive protocol. The second mode uses a manual mechanism to assemble a PayID Discovery URL from a PayID by-hand.

In 'interactive' mode, a PayID can be presented to a Webfinger-enabled service endpoint that supports PayID Discovery. The resource returns a Webfinger-compliant JavaScript Object Notation (JSON) [RFC4627] object that can be used to perform PayID Discovery as defined in section 4.1 of this document.

As an alternative, "manual" mode MAY be used to decompose a PayID into a URL, without any intermediate server interaction by simply transposing portions of a PayID URI into a URL format. This procedure is defined in section 4.2 of this document.

It should be noted that "manual" mode does not allow divergence between the string characters in a PayID URI and any corresponding PayID URL. Interactive mode, on the other hand, does allow such divergence, and is thus more powerful. For example, in manual mode, the PayID 'alice$example.com' MUST always map to the URL 'https://example.com/alice', whereas in interactive mode that same PayID URI can map to any arbitrary URL structure determined by the service provider, such as 'https://example.com/users/alice'.

Information returned via PayID Discovery might be used for direct human consumption (e.g., looking up someone's Bitcoin address), or it might be used by systems to help carry out some operation (e.g., facilitating, with additional security mechanisms, protocols to support compliance or other legal requirements necessary to facilitate a payment).

The information returned via this protocol is intended to be static in nature. As such, PayID Discovery is not intended to be used to return dynamic information like a payment account balance or the current status of a payment account.

PayID Discovery is designed to be used across many applications. Use of PayID Discovery is illustrated in the examples in Section 3 and described more formally in Section 4.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Example Usage

This section shows sample uses of PayID Discovery in several hypothetical scenarios.

3.1. PayID Discovery by a Wallet

Suppose Alice wishes to send a friend some XRP from a web-based wallet provider that Alice has an account on. Alice would log-in to the wallet provider and enter Bob's PayID (say, "bob$receiver.example.com") into the wallet UI to start the payment.

The Wallet application would first perform a WebFinger query looking for the PayID Discovery service provider, like this:

```
 GET /.well-known/webfinger?
       resource=payid%3Abob%24receiver.example.com
       HTTP/1.1
 Host: receiver.example.com
```

The server might respond like this:

```
     HTTP/1.1 200 OK
     Access-Control-Allow-Origin: *
     Content-Type: application/jrd+json

     {
       "subject" : "payid:bob$receiver.example.com",
       "links" :
       [
         {
           "rel": "https://payid.org/ns/payid-uri-template/1.0",
           "template": "https://receiver.example.com/users/{acctpart}"
         }
       ]
     }
```

Alice's wallet then uses the URL template found in the "template" property to assemble the specified PayId URL, "https://receiver.example.com/users/bob".

Per [RFC7033], Webfinger requests can be filtered by using a "rel" parameter in the Webfinger request. Because support for the "rel" parameter is not required nor guaranteed, the client must not assume the "links" array will contain only the link relations related to PayID Discovery.

3.2. PayID Discovery with Default Template

Suppose Alice, as in the example above, wishes to send a friend some XRP from a web-based wallet provider that Alice has an account on. However, in this example, let's assume that the PayID Alice is wanting to pay doesn't support "interactive" PayID discovery (i.e., the receiver's server doesn't support Webfinger).

Alice would log-in to her wallet provider and enter Bob's PayID (say "bob$receiver.example.com") to make a payment.

The Wallet application would first attempt a WebFinger query as in the example above, like this:

```
     GET /.well-known/webfinger?
           resource=payid%3Abob%24receiver.example.com&
           HTTP/1.1
     Host: receiver.example.com
```

However, in this case the "receiver.example.com" server doesn't support "interactive" PayID Discovery, so the server responds like this:

```
     HTTP/1.1 404 NOT FOUND
```

Because Alice's Wallet can utilize "manual" PayID Discovery, the wallet software merely transforms "bob$receiever.example.com" into the URL "https://receiver.example.com/bob". Alice's wallet then uses that URL to continue making a PayID payment.

It should be noted that "manual" mode does not allow the PayID URI to diverge from the underlying URL returned via PayID Discovery. Because of this, "interactive" PayID Discovery is generally preferred.

4. PayID Discovery Protocol

The PayID Discovery protocol is used to request information about an entity identified by a PayID URI.

When successful, PayID Discovery always yields a PayID URL, which is a URI as defined by [RFC3986] using the 'https' scheme defined in section 2.7.2 [RFC7230]. A PayID URL can be used for any purposes outside the scope of this document.

PayID Discovery is performed using one of two modes: "interactive" or "manual." Clients MUST attempt "interactive" mode first. If that

mode fails to yield a PayID URL, then "manual" mode MAY be used as an alternative discovery mechanism.

4.1. Interactive Mode

Interactive PayID Discovery is broken up into a series of steps, each of which is defined in more detail below. The following is a visual representation of the protocol flow:

```
                       +--------------------------+
                       |        PayID URI         |
                       |    alice$example.com     |
                       +--------------------------+
                                     |
                                     v
                       +--------------------------+
                       |         Assemble         |
                       |   PayID Discovery URL    |
                       +--------------------------+
                                     |
                                     v
                       +--------------------------+
                       |          Query           |
                 +-----|   PayID Discovery URL    |-----+
                 |     +--------------------------+     |
                 |                                  Success
                 |                                      |
                 |                                      v
                 |                        +---------------------------+
             Failure                      |   Parse PayID Metadata    |
                 |                        +---------------------------+
                 |                                      |
                 |                                      v
                 |                        +---------------------------+
                 |                        | Select PayID URI Template |
                 |                        +---------------------------+
                 |                                      |
                 v                                      v
   +--------------------------+           +---------------------------+
   |  Manual PayID Discovery  |           |    Assemble PayID URL     |
   +--------------------------+           +---------------------------+
                 |                                      |
                 +---------------------+----------------+
                                       |
                                       v
                         +---------------------------+
                         |         PayID URL         |
                         | https://example.com/alice |
                         +---------------------------+
```

4.1.1. Step 1: Assemble PayID Discovery URL

PayID Discovery utilizes the Webfinger [RFC7033] specification in a narrowly defined profile.

This document defines a PayID Discovery URL as being a Webfinger resource URI where the specified resource value is a valid PayID URI [PAYID-URI].

For example, the PayID Discovery URL for alice$example.com is

```
 https://example.com/.well-known/webfinger?resource=payid%3Abob%24example.com
```

4.1.2. Step 2: Query PayID Discovery URL

A Webfinger query MUST be performed against the PayID Disovery URL, as described in section 4.2 of Webfinger.

In response, the WebFinger resource returns a JSON Resource Descriptor (JRD) as the resource representation to convey information about the requested PayID.

If the Webfinger endpoint returns a non-200 HTTP response status code, then interactive PayID Discovery is considered to have failed. Clients MAY attempt to assemble a PayID URL using "manual" mode as defined in section 4.2.1 of this document.

4.1.3. Step 3: Parse PayID Metadata

If the PayID Discovery server returns a valid response, the response will contain one or more of the JRDs defined in section 5 of this document.

If any of the JRDs contain a 'rel' value that represents a PayID URL Template, then that template value MUST be used in the next protocol step.

Failing the above, if the 'rel' value of any JRDs represents a PayID Discovery URL, then that URL MUST be used in step 2 above, repeated recursively if needed, until a valid PayID URI Template is obtained. That URI Template value MUST be used in the next protocol step.

4.1.4. Step 4: Assemble PayID URL

A PayID URL is constructed by applying the PayID URI to the PayID URI Template string obtained in the step above. The PayID URI template MAY contain a URI string without any variables to represent a host-level PayID URL that is identical for every PayID URI on a particular host.

For example, a PayID Discovery endpoint that only supports a single account might use a URI template string with no variables, like this:

```
   {
     "rel": "https://payid.org/ns/payid-uri-template/1.0",
     "template": "https://example.com/alice"
   }
```

The result of this step is the PayID URL. Once obtained, PayID Discovery is considered to have completed successfully.

4.1.4.1. Template Syntax

This specification defines a simple template syntax for PayID URI transformation. A template is a string containing brace-enclosed ("{}") variable names marking the parts of the string that are to be substituted by the corresponding variable values.

This specification defines a one variable - "acctpart" - which corresponds to the 'acctpart' of a PayID URI as defined in [PAYID-URI].

When substituting the 'acctpart' value into a URI 'path' as defined by [RFC3986], values MUST NOT be percent or otherwise encoded because the 'acctpart' value of a PayID URI always conforms to the character set allowed by paths in [RFC3986].

However, before substituting template variables into a URI 'query' part, values MUST be encoded using UTF-8, and any character other than unreserved (as defined by [RFC3986]) MUST be percent-encoded per [RFC3986].

Protocols MAY define additional variables and syntax rules, but MUST NOT change the meaning of the 'acctpart' variable. If a client is unable to successfully process a template (e.g., unknown variable names, unknown or incompatible syntax), the JRD SHOULD be ignored.

The template syntax ABNF is as follows:

```
uri-char     =  ( reserved / unreserved / pct-encoded )
var-char     =  ALPHA / DIGIT / "." / "_"
var-name     =  %x61.63.63.74.70.61.72.74 / ( 1*var-char ) ; "acctpart" or
                                                           other names
variable     =  "{" var-name "}"
PAYID-URI-Template =  *( uri-char / variable )
```

For example:

```
    Input:    alice$example.org
    Template: https://example.org/{acctpart}
    Output:   https://example.org/alice
```

4.2. Fallback Mode

If "Interactive" mode is not supported or otherwise fails to yield a PayID URL, then a PayID URL MAY be assembled manually using the following predefined ruleset:

1. Decompose the PayID URI into its component parts, per [PAYID-URI], capturing the 'acctpart' and 'host' values.

2. Using the 'acctpart' and 'host', assemble a URL by substituting each value into the following string using no special encoding or

other character adjustments: "https://{host}/{acctpart}".

For example:

```
 Input:    bob.primary$example.org
 Output:   https://example.org/bob.primary
```

The resulting URL is a PayID URL.

4.2.1. Fallback Assembly Flow

The following is a visual representation of the Fallback Assembly protocol flow:

```
+--------------------------+
|        PayID URI         |
|    alice$example.com     |
+--------------------------+
              |
              v
+--------------------------+
|Manual PayID URL Assembly |
+--------------------------+
              |
              v
+---------------------------+
|         PayID URL         |
| https://example.com/alice |
+---------------------------+
```

5. PayID Discovery JRDs

This document defines two JRDs that conform to section 4.4 of the Webfinger RFC.

5.1. JRD for PayID Discovery URL

This type of JRD can be used to represent a URL that is a PayID Discovery URL. This is useful for delegating PayID Discovery to another service endpoint:

o 'rel': "https://payid.org/ns/payid-discovery-url/1.0"

o 'href': A PayID Discovery URL that clients can dereference to perform interactive PayID Discovery.

The following is an example of a JRD that indicates a PayID Discovery URL:

```
 {
   "rel": "https://payid.org/ns/payid-discovery-url/1.0",
   "href": "https://delegate.example.com/.well-known/webfinger?resource=
            payid%3Aalice%24example.com"
 }
```

5.2. JRD for PayID URI Template

This type of JRD can be used to represent a URL that is a PayID URL Template.

- 'rel': "https://payid.org/ns/payid-uri-template/1.0"
- 'template': A PayID URI Template

The following is an example of a JRD that indicates a PayID URI Template:

```
{
  "rel": "https://payid.org/ns/payid-uri-template/1.0",
  "template": "https://example.com/{acctpart}"
}
```

6. Security Considerations

Various security considerations should be taken into account for PayID Discovery.

Among other resource, consult section 9 of [RFC7033] and section 7 of [RFC3986] for important security considerations involved in PayID Discovery.

6.1. Hosted PayID Discovery Services

As with most services provided on the Internet, it is possible for a domain owner to utilize "hosted" WebFinger services. Consult section 7 of [RFC7033] for considerations that could apply to both "manual" and "interactive" PayID Discovery when hosted by a third-party.

6.2. Cross-Origin Resource Sharing (CORS)

PayID Discovery resources might not be accessible from a web browser due to "Same-Origin" policies. See section 5 of [RFC7033] for CORS considerations that apply to both "manual" and "interactive" PayID Discovery modes.

6.3. Access Control

As with all web resources, access to the PayID Discovery resource could require authentication. See section 6 of [RFC7033] for Access Control considerations that could apply to both "manual" and "interactive" PayID Discovery modes.

7. IANA Considerations

7.1. New Link Relation Types

This document defines the following Link relation types per [RFC7033]. See section 3 for examples of each type of Link.

7.1.1. PayID Discovery URL

o Relation Type ('rel'): "https://payid.org/ns/payid-discovery-url/1.0"

o Media Type: "application/jrd+json"

o Description: PayID Discovery URL, version 1.0

7.1.2. PayID Discovery URI Template

o Relation Type ('rel'): "https://payid.org/ns/payid-uri-template/1.0"

o Media Type: "application/jrd+json"

o Description: PayID Discovery URI Template, version 1.0

8. Acknowledgments

This document was heavily influenced by, and builds upon, Webfinger [RFC7033] (adapted for a payments use-case) as well as the supporting RFCs that it relies upon and that influenced it, especially [RFC5988] and [RFC6415]. The author would like to acknowledge the contributions of everyone who worked on those and any related specifications.

In addition, the author would like to acknowledge everyone who provided feedback and use-cases for this derivative specification.

9. References

9.1. Normative References

[PAYID-URI]
Fuelling, D., "The 'payid' URI Scheme", n.d., <https://tbd.example.com/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, DOI 10.17487/RFC4627, July 2006, <https://www.rfc-editor.org/info/rfc4627>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012,

<https://www.rfc-editor.org/info/rfc6570>.

[RFC7033] Jones, P., Salgueiro, G., Jones, M., and J. Smarr, "WebFinger", RFC 7033, DOI 10.17487/RFC7033, September 2013, <https://www.rfc-editor.org/info/rfc7033>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

9.2. Informative References

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <https://www.rfc-editor.org/info/rfc5988>.

[RFC6415] Hammer-Lahav, E., Ed. and B. Cook, "Web Host Metadata", RFC 6415, DOI 10.17487/RFC6415, October 2011, <https://www.rfc-editor.org/info/rfc6415>.

9.3. URIs

[1] https://payid.org/

[2] mailto:payid@ripple.com

Author's Address

David Fuelling
Ripple
315 Montgomery Street
San Francisco, CA 94104
US

Phone: -----------------
Email: fuelling@ripple.com
URI: https://www.ripple.com