Packetizer™

WebFinger Server

Version: 1.3

October 21, 2013

Contents

Legal Information

1 Introduction

WebFinger is a HTTP-based protocol for discovering information about people and things on the Internet. Given a Universal Resource Identifier (URI), such as an email address or account identifier, one can find the address of a person's blog, address card (e.g., vCard or xCard), avatar, or other useful information. The response to a WebFinger query is a JSON object that is suitable for processing by the requesting application.

This server software implements all features specified in the WebFinger specification (RFC 7033).

2 Technology Employed

The WebFinger Server is written entirely in the popular scripting language Perl, meaning that it should be easily ported to any Linux machine without any effort. It was written using Perl 5.14, though it should work with older versions of Perl. There was no intent or desire to use "cutting edge" Perl features, as the primary objective was to ensure portability of code and ease of system recovery in the event of a failure.

The server relies on MySQL for data storage. It might be possible to utilize other databases, such as SQLite, but no effort was made and no testing has been performed. MySQL was selected for performance reasons and for the fact that tools like phpMyAdmin and MySQL Workbench make it simple to administer.

The server was written for the Apache web server, though it might work for other web servers. However, no other web servers have been targeted or tested. There are no new or advanced features required from the Apache web server, so virtually any deployed Apache server should work with the server.

3 Server Software Components

The server software is comprised of a main CGI script called "webfinger.cgi" and two library files, one that provides database connectivity functions and one used to configure parameters.

The WebFinger Server software package contains the following files:

Directory	Tool	Description
cgi/		
	webfinger.cgi	This is the main WebFinger Server application. It will provide a response to queries for the well-known resource "webfinger".
config/		
	apache.config	This file shows how one can use re-write rules so that requests that come to the server might be directed to the appropriate location. This is just a sample configuration and you should modify this file for your environment.
	webfinger.sql	This contains the SQL statements to create the database tables used by the server. Note that all table names start with

		“webfinger_” to avoid name collisions, should you wish to use an existing database.
docs/		
	WebFinger Server.pdf	This is the document you are reading.
lib/		
	database.pl	This is a library that contains the functions to connect and disconnect from the database.
	webfinger_config.pl	This contains a set of global configuration variables. This script must be modified to contain values that are appropriate for the environment.

4 Installing the Software

4.1 Extract the Server Software

Extract the installation file to create the file structure described in the previous section. The file is provided as a .tgz file that can be easily extracted using the following command on Linux:

```
$ tar -xzvf /path/to/webfinger_distribution.tgz
```

4.2 Install MySQL and Create the Database Tables

Installing MySQL the first time will require a bit of work, but take notes and save your configuration files. Installing it a second time will be a snap, should you migrate to another server, for example. This document will not go into detail on MySQL installation, maintenance, tuning, etc. There are numerous resources on the Internet that do that already. Documentation abounds here: http://dev.mysql.com/doc/index.html.

Once you have the MySQL server running, you either need to create a new database or select an existing database. Let’s assume you create a database called “webfinger”. Once the database is created, use the following command to create the tables used by the server:

```
$ mysql webfinger < config/webfinger.sql
```

If you do not already have a database “user” account defined that the WebFinger server can use, be sure to create one and give that user usage rights to the webfinger-related tables. You may grant additional privileges, but “usage” rights are needed at a minimum.

4.3 Putting the CGI Script in Place

Where you put the CGI script is not so important, but a convenient location is in the htdocs/.well-known directory on the server. If you have no other .well-known services on the server, then you may need to create the .well-known directory.

You next need to make the Apache configuration changes. Look at the sample configuration for guidance (in the config/apache.conf file). The configuration example does a couple of important things:

1. Any attempt to access the .cgi script directly is met with a 404 response code
2. A request to /.well-known/webfinger invokes the webfinger.cgi script and passes the query string

The configuration lines are designed to go either in an .htaccess file or in the <VirtualHost> section of the Apache configuration file.

Note that the WebFinger specification states that WebFinger is required to be used only over TLS connections. You must install the script and configuration lines on servers where TLS is supported. On most Apache servers, that might mean adding lines to the ssl.conf file as appropriate, though use of .htaccess is certainly possible and popular with generic hosting services.

4.4 Populating the Database

The server does not come with software for populating the database. This must be done manually using the 'mysql' command or using tools like phpMyAdmin. A complete explanation of every table and every field is found in Section 6.

Start by inserting a "resource". A "resource" is a person or thing about which you would like others to be able to discover information.

For example, let's say there is a user named "bob" with an email (mailto:) address and an account (acct:) URI in the domain "example.com". One would start by inserting an account URI into the "webfinger_resources" table. The row might have the following data elements after the insertion:

skey	resource	name
1	acct:bob@example.com	Bob Smith

The skey field is used to relate data about Bob to properties and links also defined for Bob. The "name" column is there for the administrator's benefit. It is never returned in a query to the server.

Bob has an email address, which is really just an alias for his account. So, insert a row into the "webfinger_aliases" table for Bob's email address. The data record would look something like this:

skey	pkey	alias
1	1	mailto:bob@example.com

The "skey" field in the "webfinger_aliases" table is the serial key row for that table. The "pkey" refers to the "parent key", which is the field webfinger_resources.skey. All tables are related in this way, as described in the database schema section.

With the alias in place, a user issuing a query to the server for either acct:bob@example.com or mailto:bob@example.com will receive a response back that contains substantially the same information. The only difference is that the server will make changes to the "aliases" array returned as appropriate. Specifically, if one queries an alias like "mailto:bob@example.com", the alias is actually the "subject" of the request and is therefore removed from the aliases array. However, the primary resource identifier "acct:bob@example.com" is inserted into the aliases array.

A user might also have defined "properties". A property might be the person's name or other useful information. A table of commonly used "properties" will be maintained here: http://www.packetizer.com/webfinger/properties.html.

Let's assume Bob's name is published via a property record, both in Bob's default language and simplified Chinese. The entries in the webfinger_resource_properties table might look like this:

skey	pkey	type	value
1	1	http://packetizer.com/ns/name	Bob Smith
2	1	http://packetizer.com/ns/name#zh-CN	鲍勃 • 史密斯

Next, there would be a set of links that point to information about Bob. Let's assume Bob has a blog located at http://blogs.example.com/bob/ called "The Magical World of Bob" in English and "Le Monde Magique de Bob" in French. To publish that fact via WebFinger, one would define the link to the blog using the "webfinger_links" table. The entry for Bob's Blog might look like this:

skey	pkey	order	relation	type	template	href
1	1	1	http://packetizer.com/rel/blog	text/html	N	http://blogs.example.com/bob/

As before, the pkey relates to the resource record for Bob. The order column indicates the order in which records should be placed into the "links" array returned to the requesting client. The "relation" column specified the link relation type. The above type refers to a blog. The "type" column (which is optional) refers to the media type expected to be returned by the resource. Finally, the "href" column provides the URL to Bob's blog.

Recall that Bob has two titles for his blog. These titles may be inserted into the "webfinger_link_titles" table as in the following:

skey	pkey	lang	title
1	1	en-US	The Magical World of Bob
2	1	fr	Le Monde Magique de Bob

Here, pkey refers to the "webfinger_links.skey" value, which allows association of these titles with the parent link.

WebFinger also defines "properties" for links. The link properties table structure is exactly the same as the properties table for resources. For example usage of link properties, please see the documentation on this site: http://www.packetizer.com/webfinger/.

That's about it. It's pretty simply to add additional resources or links to the database.

4.5 Modify the webfinger_config.pl File

The lib/config.pl file contains a lot of system-wide configuration parameters that must be set before you try to use the server. All of the configuration parameters are defined below. Note that all of the variables start with "$main::". This is Perl syntax to indicate that the variable is global. Do not change that.

Parameter	Description
database_user_id	This is the MySQL database "user" identifier that the server should use.
database_password	This is the MySQL database password associated with the above user ID.
database_name	This is the name of the MySQL database to access.
database_server	This is the hostname where the MySQL database server is running. By default, this value is set to "localhost", since it is assumed the MySQL database will be local. However, the MySQL database could be installed on an entirely separate machine.
database_use_ssl	If set to 1, then SSL will be used for database connections. A value of 0 indicates that SSL is not used.
cors_policy	This is the CORS policy advertised in replies to the client. The default value is "*", but may be changed. For more information about CORS, see http://www.w3.org/TR/cors/.
status_messages	This is a hash of status codes and associated terse messages. Generally, this should not be modified.
error_messages	This is a hash of status codes and associated verbose messages. You may wish to modify these to provide better description or to return a message in a different language.

4.6 Putting the files from the "lib" directory in place

Where you place the files from the "lib" directory is not important, but the Perl script has to be able to find them when executed. Let's assume that you have a directory for storing files related to your web site called "example.com". Inside that directory, you might have this structure:

```
example.com/
       htocs/
       cgi-bin/
       lib/
```

Given this structure, you could put the "lib" files into the above "lib" directory. Apache needs to be told where to find the files. To do that, set the PERL5LIB environment variable in the .htaccess file or inside the <VirtualHost> directive, like this:

```
    SetEnv PERL5LIB /path/to/example.com/lib/
```

5 Using the WebFinger and Host Metadata Server

Once installed and configured, you can start using the server immediately after restarting the web server. You can perform some simple tests using tools like curl:

```
$ curl https://example.com/.well-
known/webfinger?resource=acct:bob@example.com
```

Of course, you should change the URLs to be appropriate for your installation.

6 Database Schema

There are a six database tables defined for use by the server software. They are each presented in this section with an explanation of each table and column. Every text field is defined to be a UTF-8 string. All unsigned integers (uint) are 64 bits in length.

6.1 "webfinger_resources" Table

The "webfinger_resources" table defines all of the people and things that should be discoverable when the server is queried.

Column Name	Type	Description
skey	uint	A unique serial number assigned to each resource record
resource	varchar(255)	The URI of the resource
name	varchar(255)	A name the server admin would like to assign to the resource to make data management easier.

6.2 "webfinger_resource_properties" Table

The "webfinger_resource_properties" table defines properties related to the host or to a specific resource.

Column Name	Type	Description
skey	uint	A unique serial number assigned to each resource property record
pkey	uint	This is the serial key for the resource (webfinger_resources.skey)
type	varchar(255)	An identifier that identifies the property type
value	varchar(255)	The value of the property

6.3 "webfinger_aliases" Table

The "webfinger_aliases" table defines aliases for a specific resource.

Column Name	Type	Description
skey	uint	A unique serial number assigned to each alias record
pkey	uint	This is the serial key for the resource (webfinger_resources.skey)
alias	varchar(255)	This is the alias URI for the specified resource

6.4 "webfinger_links" Table

The "webfinger_links" table defines links related to the host or to a specific resource.

Column Name	Type	Description
skey	uint	A unique serial number assigned to each link record
pkey	uint	This is the serial key for the resource (webfinger_resources.skey)
order	uint	This is the order in which links are to be placed into the array when returned to the client, with the lower values appearing first in the links array
relation	varchar(255)	This is the link relation type, such as 'copyright' or 'http://packetizer.com/rel/blog'
type	varchar(128)	This is the media type expected to be returned by the related link
href	varchar(1024)	This is the target URI for the link relation

6.5 "webfinger_link_properties" Table

The "webfinger_link_properties" table defines properties related to a specific link.

Column Name	Type	Description
skey	Uint	A unique serial number assigned to each link property record
pkey	Uint	This is the serial key for the link (webfinger_links.skey)
type	varchar(255)	An identifier that identifies the property type
value	varchar(255)	The value of the property

6.6 "webfinger_link_titles" Table

The "webfinger_link_titles" table defines titles related to a specific link.

Column Name	Type	Description
skey	uint	A unique serial number assigned to each link property record
pkey	uint	This is the serial key for the link (webfinger_links.skey)
lang	varchar(255)	The language associated with a specific title (may be "default")
title	varchar(255)	A title for the link