WebFinger Email Automatic Configuration
draft-jones-webfinger-email-autoconfig-00

Abstract

This document describes procedures for automatically configuring an
email client by using WebFinger to convey mail server configuration
and security-related information.

Status of This Memo

This Internet-Draft is submitted in full conformance with the
provisions of RFC 78 and RFC 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 https://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
text in RFCs.

This Internet-Draft will expire on January 9, 2020.

Copyright Notice

Copyright (c) 2019 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
(https://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 4.1.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.
1. Introduction

Configuring email clients manually can be a time-consuming and mundane task that can actually be fairly arduous as the number of users or clients increases. While people with technical understanding of the various protocols and settings employed can configure a single email client fairly easily, the task can prove challenging and frustrating for non-technical users.

This document defines procedures utilizing WebFinger [RFC7033] that make it trivial for end-users to configure email clients with little or no assistance and with minimal technical skill. All that is required for the user to do is enter his or her account identifier, select an email account, and provide credentials.

Following these procedures, the complexity of configuring an email client is shifted away from the end-user and pushed onto the software tools that implement these procedures and the email administrators who would have to configure the settings for each user server-side and publish the required configuration data. However, the configuration data can be generated with automated tools, meaning there would be minimal effort for the email administrators.

2. Conventions Used in This Document

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 BCP
3. Example User Scenario

To illustrate how the procedures work, assume Alice just got a new computer and installed her favorite email client. She is using services from Example Mail Corporation and she was told her account identifier is "alice@mail.example" and her password is "abc123". She actually has two different accounts with Example Mail Corporation, one for personal use and one for her business. The accounts are aptly named "Personal" and "Business".

When Alice starts her email client and adds a new email account, the client will ask her for her account identifier. She will then enter "alice@mail.example", at which point the client will initiate a WebFinger query to the host "mail.example". That HTTP [RFC7230] query will look something like this:

```
GET /.well-known/webfinger?resource=acct%3Aalice%40mail.example HTTP/1.1
Host: mail.example
```

(The backslash character '\' is there only to indicate the following line is part of the same line.)

As per the WebFinger specification, a JSON Resource Descriptor (JRD) will be returned with a number of different entries. For brevity, the following reply shown excludes most of the content that is not applicable to this example. The reply might look like this:
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/jrd+json

{
  "subject": "acct:alice@mail.example",
  "links": [
    {
      "rel": "email-autoconfig",
      "href": "https://mail-config.mail.example/alice.personal.json"
    },
    {
      "rel": "email-autoconfig",
      "href": "https://mail-config.mail.example/alice.business.json"
    }
  ],
  "titles": {
    "en-us": "Personal"
  }
}

Noting there are two "email-autoconfig" entries in the response, one named "Personal" and one named "Business", Alice's email client will prompt her to select one of the email accounts to configure to provide a password. She selects the one named "Personal", at which point the email client will issue a subsequent HTTP query to the URI associated with that account:

   GET /alice.personal.json HTTP/1.1
   Host: mail-config.mail.example

The server "mail-config.mail.example" might require her to authenticate using the password for her email account, which is why she was prompted for it when selecting the account. Once the email client satisfies whatever authentication challenge it is presented, the server will then respond with a JSON document that contains all of the configuration information necessary for the email client to self-provision her "Personal" account. That reply might look something like this:
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/jrd+json

{
  "address" : "alice.i.wonder@mail.example",
  "smtp" :
  {
    "login" : "alice.i.wonder",
    "host" : "outbound.mail.example",
    "port" : 587,
    "login-required" : true,
    "transport" : "starttls"
  },
  "imap" :
  {
    "login" : "alice.i.wonder",
    "host" : "imap.mail.example",
    "port" : 143,
    "transport" : "starttls"
  }
}

The email client would utilize this information to configure itself to access the mail server using SMTP and IMAP. The email client would reasonably assume that the account password provided is the same password used with both of those protocols. While that might not be correct, it certainly simplifies the user’s life if it is the same. However, authentication may be via a digital certificate, for example, conveyed in this response document.

4. Email Auto-Configuration Procedures

   TBD

4.1. Initiating a WebFinger Query

   TBD

4.2. Processing the JSON Resource Descriptor

   TBD

4.3. Requesting the Email Configuration Document

   TBD
5. Email Configuration Document Syntax

TBD

6. Security Considerations

TBD

7. IANA Considerations

TBD

We will need to register "email-autoconfig" in the Link Relations registry: https://www.iana.org/assignments/link-relations/link-relations.xhtml [1]

8. Acknowledgments

TBD

9. References

9.1. Normative References


9.2. Informative References


Jones & Salgueiro Expires January 9, 2020 [Page 6]
9.3. URIs

[1] https://www.iana.org/assignments/link-relations/link-relations.xhtml

Authors’ Addresses

Paul E. Jones
Cisco
7025 Kit Creek Rd.
Research Triangle Park, North Carolina 27709
USA

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

Gonzalo Salgueiro
Cisco
7025 Kit Creek Rd.
Research Triangle Park, North Carolina 27709
USA

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