Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 30, 2013

A. McMillan
Morphoss
C. Daboo
Apple Inc.

June 28, 2013

Automated Service Configuration
draft-daboo-aggregated-service-discovery-03

Abstract

This specification describes how clients can retrieve configuration for multiple services with a minimum of user-provided information, and as short as possible sequence of queries, and with a minimum of overhead for administrators of the services.

Status of This Memo

Copyright Notice

Table of Contents

```
   1.  Introduction . . . . . . . . . . . . . . . . . . . . . . . . .   3
   2.  Open Issues  . . . . . . . . . . . . . . . . . . . . . . . . .   4
   3.  Conventions Used in This Document  . . . . . . . . . . . . . .   4
   4.  Overview . . . . . . . . . . . . . . . . . . . . . . . . . . .   4
   5.  Automated Service Configuration Document Format  . . . . . . .   4
     5.1.  Extensions to the Document Format  . . . . . . . . . . . .   8
     5.2.  Service names  . . . . . . . . . . . . . . . . . . . . . .   8
   6.  Finding the Automated Service Configuration Information  . . .   8
   7.  Handling multiple, alternative services  . . . . . . . . . . .   9
   8.  Internationalization Considerations  . . . . . . . . . . . . .  10
   9.  Security Considerations  . . . . . . . . . . . . . . . . . . .  10
   10. IANA Considerations  . . . . . . . . . . . . . . . . . . . . .  10
     10.1. Link Relation Registration . . . . . . . . . . . . . . . .  10
       10.1.1.  service-configuration Link Relation Registration  . .  10
   11. Acknowledgments  . . . . . . . . . . . . . . . . . . . . . . .  11
   12. References . . . . . . . . . . . . . . . . . . . . . . . . . .  11
     12.1. Normative References . . . . . . . . . . . . . . . . . . .  11
     12.2. Informative References . . . . . . . . . . . . . . . . . .  12
   Appendix A.  Change History (to be removed prior to
                publication as an RFC)  . . . . . . . . . . . . . . .  12
   Appendix B.  Example of multiple services  . . . . . . . . . . . .  13
   Appendix C.  Example - multiple, alternative mail retrieval
                services  . . . . . . . . . . . . . . . . . . . . . .  15
   Appendix D.  Example - multiple links  . . . . . . . . . . . . . .  18
```

1. Introduction

There are currently various systems in place for discovery and configuration of individual services, but the process can often require an extensive series of requests using different protocols to discover all of the details needed to set up the various client services which an individual might use to interact with an organisation or service provider.

Consider Jason, a new employee at Example Enterprises. Jason needs to configure his e-mail program to use IMAP [RFC3501] + TLS on port 143 against mail.example.com, he needs to send mail on port 8557 via TLS+SMTP to smtp.example.com, his calendar is on port 8443 at https://caldav.example.com:8443/calendar/, and so forth. Some of these things can be discovered relatively easily, with a combination of DNS queries (including SRV lookups, certificate checking, and http requests). However, each protocol has its own requirements and settings and each has to be done separately. Whilst the client can "hide" the multiple service setup from the user, the actual implementation often requires separate code and processes to manage, making it more complex that it needs to be.

This specification defines a single protocol which will allow for retrieval of a variety of service configuration information in a single call, allowing developers to simplify the coding and user interface in client software, and in particular in multi-function client software such as a combined e-mail and calendar clients. Configuration is retrieved from a single document on a server, improving performance over per-service configuration mechanisms that require multiple network operations. In addition, complex dependencies between different services can be easily represented, so that, for example, some services can be prioritized over others, or grouped together by "logical" function. Further, rich information about each service can be included, such as details about required transport layer security or authentication.

This protocol is intended to provide a simple one-way retrieval of configuration information, with the data flow from service provider to client, building on successful technologies such as HTTP and JSON to accomplish its goals. As such it differs from other "client configuration" orientated protocols, such as ACAP [RFC2244], which have not had much deployment success due to complexity introduced by having a a two-way exchange of data, and the resulting need to keep multiple clients synchronized as changes occur.

2. Open Issues

1. Is it OK to embed certificate details for the actual services or a root certificate?

3. Conventions Used in This Document

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 [RFC2119].

4. Overview

The following outlines the steps a client carries out to setup multiple services for a user:

1. The client software is expected to capture a user identifier and domain name (possibly entered in the form of an email address) from the user, and possibly authentication information. e.g., 'cyrus@example.com'.

2. The client uses the WebFinger [I-D.ietf-appsawg-webfinger] protocol to request a link relation with the value "service-configuration", using the identifier supplied by the user, following any redirects and responding to any authentication challenges.

3. The client retrieves a JSON [RFC4627] document conforming to the format described in Section 5 from the target of the link relation returned in Step #2. The client parses this document to extract information about the available services. At that point it can either present a list of services to the user, so that they can decide exactly what they want setup, or it can automatically setup services for all those it supports.

5. Automated Service Configuration Document Format

The automated service configuration document is an JSON [RFC4627] document. The document contains a single object with two members representing two pieces of information: overall service provider information (e.g., name, icon "badge", contact information), and a list of each service supported. Each service will contain some information common to each type of service, and then information specific to each service.

The JSON document format is defined here using the syntax in [I-D.newton-json-content-rules]. An example of such an automated service configuration document for some common services is shown in

Appendix B.

JSON Content Rules for the JSON document returned for a "capabilities" action request.

```
; root object

root {
  provider,
  entries
}

; ----- provider -----

; Contains information describing the service provider, that can be
; used by clients to "group" individual services together under a
; common name or section when presenting details to the user.
provider "provider" {
  provider_name,
  ?description,
  ?image,
  ?contact,
  ?manage,
  ?password_reset,
  ?ttl
}

; The name for the service provider.
provider_name "name" : string

; The description of the service provider.
description "description" : string

; A URI for an image that can be used as an "icon" for the service
; provider. The URI SHOULD be an http or https URI and clients
; SHOULD use standard HTTP Accept header behavior to request an
; appropriate image format from the server (see Section 14.1 of
; [RFC2616]). The image SHOULD NOT exceed a size of 128 x 128
; pixels.
image "image" : uri

; Contact information for the service provider.
contact "contact" {
  email / (?email, uri)
}

; An email address that can be used to contact the service
; provider.
```

```
   email "email" : email

   ; A URI for a webpage providing information about the service
   ; provider.
   url "url" : uri

   ; A URI for a webpage where a user can manage details of their
   ; account.
   ; e.g., a place where users can go to add additional (possibly
   ; payment required) services.
   manage "manage" : uri

   ; A URI for a webpage where a user can change their account
   ; password.
   password_reset "password-reset" : uri

   ; The minimum interval in seconds which clients SHOULD wait
   ; before re-fetching the document to check for changes.
   ttl "ttl" : integer

   ; ----- entries -----

   ; List of services.
   entries "entries" [ *entry ]

   entry {
     name,
     service,
     ?(group, priority),
     uri / (host, ?port),
     ?tls,
     ?auth,
   }

   ; A description for the service.
   name "name" : string

   ; The service type. See below for details of the value used.
   service "service" : string

   ; Identifies the nature of the service to allow similar services to
   ; be grouped together.
   group "group" : string

   ; Identifies the nature of the service to allow similar services to
   ; be grouped together.
   priority "priority" : integer 1
```

```
   ; The URI used to contact the server providing the service.
   uri "uri" : uri

   ; The hostname of the server providing the service.
   host "host" : string

   ; The network port number of the server providing the service.
   port "port" : integer 0..65535

   ; Provides detail of transport layer security to be used with the
   ; service.
   tls "tls" {
     ?required,
     ?at_start,
     ?certificates
   }

   ; Indicates that clients MUST use transport layer security when
   ; connecting to the server providing the service.
   required "required" : boolean

   ; Indicates that clients MUST initiate TLS immediately upon
   ; connecting to the server rather than using an "in-protocol"
   ; upgrade mechanism.
   at_start "at-start" : boolean

   ; List of certificates.
   certificates "certificates" [ *certificate ]

   ; Details about the TLS certificate the server will use. Clients
   ; MAY use the specified certificate information to validate any TLS
   ; connection to the server, otherwise existing rules for the target
   ; protocol are used.
   certificate "certificate" {
     cert_name /
     fingerprint /
     public_key
   }

   ; The name of the certificate.
   cert_name "name" : string

   ; The fingerprint of the certificate.
   fingerprint "fingerprint" : string

   ; The fingerprint of the certificate.
   public_key "public-key" : string
```

```
; List of authentication methods to use in server preferred order.
; If the protocol supports SASL [RFC4422] then this is a list of
; SASL authentication mechanisms, otherwise it is a protocol
; specific list of names. In either case, clients MUST NOT use
; a mechanism that is not advertised in this list.
auth "auth" [ *string ]
```

5.1. Extensions to the Document Format

Additional members can be added to the JSON document root, "provider" or "entries" objects with the following rules:

1. Standards based members MUST be defined in an RFC and registered with IANA. TBD - precise details of this and IANA registry setup.

2. Member names that include a prefix of the form "{...}", where the contents of the curly braces is a vendor id, are considered to be vendor specific private extensions which do not require registration. TBD nature of vendor id.

Clients SHOULD ignore all extension member elements that they are unable to process.

5.2. Service names

The "service" member in the "entry" object is used to convey an identifier for the type of service being described. This can have one of two forms:

1. An identifier from the IANA ports registry defining a service type.

2. Identifiers that include a prefix of the form "{...}", where the contents of the curly braces is a vendor id, are considered to be vendor specific private service type. TBD nature of vendor id.

6. Finding the Automated Service Configuration Information

This specification makes use of the WebFinger [I-D.ietf-appsawg-webfinger] protocol to allow a client to find a link to the automated service configuration document corresponding to an identifier supplied by a user. This specification registers the "service-configuration" link relation type for use with WebFinger, to identify the links for automated service configuration documents. When the client makes a WebFinger request, it SHOULD use the "rel" query parameter, defined in Section 4.3 of

[I-D.ietf-appsawg-webfinger], to ensure the relevant link relations are returned. The URI referenced by the links MUST be an HTTPS [RFC2818] URI, and MUST point to a resource that the client can use to retrieve the automated service configuration document for the site.

Servers can return multiple "service-configuration" links in the webfinger response. Clients SHOULD treat each link as separate, complementary services available to the user.

When requesting automated service configuration documents, clients MUST include a URI query parameter "id" set to the user identifier entered by the user. Clients MUST handle HTTP redirects on the link URI, but MUST NOT allow a redirect to an insecure URI. When responding to the request, the server MUST tailor the automated service configuration document for the user making the request and MUST require HTTP authentication by that user before returning the document.

Clients SHOULD cache the document for a period of time no less than the value of the "ttl" member in the "provider" object, or for a minimum of 24 hours of no "ttl" member is present.

7. Handling multiple, alternative services

The "group" and "priority" members of an entry provide a way for a service provider to distinguish multiple services of the same type, as well as allow the client to select the most appropriate service when several alternatives exist.

For example, consider the case of a service provider supporting two separate email retrieval services, one the "primary" account, and the other for "internal" messaging only. It is expected that clients configure accounts for both services. Each service also offers either IMAP [RFC3501] or POP3 [RFC1939] as an email retrieval protocol. In this case the automated service configuration document would contain four entry items: two describing an IMAP service and two describing a POP3 service. Each entry would contain a "group" member that groups one IMAP and one POP3 service together for each of the "primary" and "internal" account groups. Each entry would also contain a "priority" member indicating the service providers preference for clients to use either IMAP or POP3. An example of such an automated service configuration document is shown in Appendix C.

When a client retrieves and processes such a document, it would first group services based on the SD:application value. For each group, it iterates over the list of entries in the group, ordered by SD:

priority values, and configures an account for the first one it finds with an SD:service that it supports.

8. Internationalization Considerations

Some elements of the automated service configuration document can contain human readable text that clients might choose to present to a user. Clients SHOULD use the Accept-Language header behavior described in Section 14.4 of [RFC2616] to ensure the server can return a document suitable for the user's chosen language. Servers SHOULD support variations of the automated service configuration document based on language, returning the appropriate variation in response to client requests.

9. Security Considerations

When using WebFinger to discover a server hosting the automated service configuration document, a malicious attacker with access to the DNS server data, or able to get spoofed answers cached in a recursive resolver, can potentially cause clients to connect to a server hosting a bogus automated service configuration document with service data chosen by the attacker. In the absence of a secure DNS option, clients SHOULD check that the target FQDN returned in the link relation URI record matches the original service domain that was queried. If the target FQDN is not in the queried domain, clients SHOULD verify with the user that the link URI target FQDN is suitable for use before executing any connections to the host.

HTTP requests for the automated service configuration document MUST be performed via TLS. Clients MUST use the procedure outlined in Section 4.3 of [RFC6125] to verify the service.

10. IANA Considerations

10.1. Link Relation Registration

This document defines a "service-configuration" link relation type using the registration procedure and template from Section 6.2 of [RFC5988].

10.1.1. service-configuration Link Relation Registration

Relation Name: service-configuration

Description: Refers to an automated service configuration document

Reference: This RFC.

11. Acknowledgments

The authors would like to thank the following individuals for contributing their ideas and providing feedback for writing this specification: Andrew Biggs, Mike Douglass, Joe Hildebrand, Paul Jones, Markus Lanthaler, Matt Miller, Stepan Potys, and Peter Saint-Andre

The authors would also like to thank CalConnect, The Calendaring and Scheduling Consortium, for advice with this specification.

12. References

12.1. Normative References

[I-D.ietf-appsawg-webfinger] Jones, P., Salgueiro, G., and J. Smarr, "WebFinger", draft-ietf-appsawg-webfinger-14 (work in progress), May 2013.

[I-D.newton-json-content-rules] Newton, A., "A Language for Rules Describing JSON Content", draft-newton-json-content-rules-01 (work in progress), January 2013.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] 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.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC4422] Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627,

	July 2006.
[RFC5988]	Nottingham, M., "Web Linking", RFC 5988, October 2010.
[RFC6125]	Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

12.2. Informative References

[RFC1939]	Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, May 1996.
[RFC2244]	Newman, C. and J. Myers, "ACAP -- Application Configuration Access Protocol", RFC 2244, November 1997.
[RFC3501]	Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

Appendix A. Change History (to be removed prior to publication as an RFC)

Changes in -03:

1. Fix examples to properly show request/response headers.

2. Title change.

3. Added text to the Introduction to differentiate this from other protocols.

4. Switched to WebFinger instead of SRV.

Changes in -02:

1. Switched to JSON document format.

Changes in -01:

1. Renamed various elements for clarity.

2. Added an SD:manage element.

3. Added a section on handling of multiple, alternative services, together with a second appendix example.

Appendix B. Example of multiple services

First comes the WebFinger request to retrieve the appropriate link relation.

```
>> Request <<

GET /.well-known/webfinger?
  resource=acct:cyrus@example.com&
  rel=service-configuration HTTP/1.1
Host:example.com:443
Authorization: basic QmFzZTY0IGlzIGVhc3kgdG8gZGVjb2Rl

>> Response <<

HTTP/1.1 200 OK
Date: Wed, 20 Feb 2013 09:32:12 GMT
Content-Type: application/jrd+json
Content-Length: xxx

{
  "subject" : "acct:cyrus@example.com",
  "links" :
  [
    {
      "rel" : "service-configuration",
      "type" : "application/json",
      "href" : "https://example.com/service-config"
    }
  ]
}
```

Next is the retrieval of the automated service configuration document using the URI from the WebFinger link relation.

```
>> Request <<

GET /service-config?id=cyrus@example.com HTTP/1.1
Host:example.com:443
Authorization: basic QmFzZTY0IGlzIGVhc3kgdG8gZGVjb2Rl
```

```
>> Response <<

HTTP/1.1 200 OK
Date: Wed, 20 Feb 2013 09:32:12 GMT
Content-Type: application/json
Content-Length: xxx

{
  "provider" : {
    "name" : "Super-duper ISP",
    "description" : "Super-duper ISP is the home for all your data.",
    "contact" : {
      "email" : "superduper@example.com",
      "uri" : "http://www.example.com"
    },
    "manage" : "http://www.example.com/myaccount.html",
    "ttl" : 2592000
  },

  "entries" : [
    {
      "name" : "Corporate Mail",
      "service" : "imap",
      "group" : "mail-access-1",
      "priority" : 2,
      "uri" : "imap:imap.example.com",
      "tls" : {
        "required" : true
      },
      "auth" : ["CRAM-MD5"]
    },

    {
      "name" : "Corporate Mail",
      "service" : "pop3",
      "group" : "mail-access-1",
      "priority" : 1,
      "host" : "mail.example.com",
      "port" : 110,
      "tls" : {
        "required" : true
      },
      "auth" : ["CRAM-MD5"]
    },

    {
      "name" : "Corporate Mail",
      "service" : "submission",
```

```
         "host" : "mail.example.com",
         "port" : 587,
         "tls" : {
           "required" : true
         },
         "auth" : ["CRAM-MD5"]
       },

       {
         "name" : "Corporate Calendar",
         "service" : "caldav",
         "uri" : "https://calendar.example.com",
         "tls" : {
           "required" : true,
           "at-start" : true
         },
         "auth" : ["Digest"]
       },

       {
         "name" : "Corporate Contacts",
         "service" : "carddav",
         "uri" : "https://contacts.example.com",
         "tls" : {
           "required" : true,
           "at-start" : true
         },
         "auth" : ["Digest"]
       }
     ]
   }
```

Appendix C. Example - multiple, alternative mail retrieval services

First comes the WebFinger request to retrieve the appropriate link relation.

```
>> Request <<

GET /.well-known/webfinger?
  resource=acct:cyrus@example.com&
  rel=service-configuration HTTP/1.1
Host:example.com:443
Authorization: basic QmFzZTY0IGlzIGVhc3kgdG8gZGVjb2Rl

>> Response <<

HTTP/1.1 200 OK
Date: Wed, 20 Feb 2013 09:32:12 GMT
Content-Type: application/jrd+json
Content-Length: xxx

{
  "subject" : "acct:cyrus@example.com",
  "links" :
  [
    {
      "rel" : "service-configuration",
      "type" : "application/json",
      "href" : "https://example.com/service-config"
    }
  ]
}
```

Next is the retrieval of the automated service configuration document using the URI from the WebFinger link relation. This example shows two different email services: "Primary Mail" (which has both IMAP4 and POP3 services available), and "Internal Mail" (which additionally has a private "webmail" service available). An extension member is also specified for the "webmail" service.

```
>> Request <<

GET /service-config?id=cyrus@example.com HTTP/1.1
Host:example.com:443
Authorization: basic QmFzZTY0IGlzIGVhc3kgdG8gZGVjb2Rl

>> Response <<

HTTP/1.1 200 OK
Date: Wed, 20 Feb 2013 09:32:12 GMT
Content-Type: application/json
Content-Length: xxx

{
```

```
     "provider" : {
       "name" : "Mail Agrregator ISP",
       "description" : "Primary and internal email services.",
       "contact" : {
         "email" : "emails@example.com",
         "uri" : "http://www.example.com"
       },
       "manage" : "http://www.example.com/myaccount.html",
       "ttl" : 2592000
     },

     "entries" : [
       {
         "name" : "Primary Mail",
         "service" : "imap",
         "group" : "primary",
         "priority" : 2,
         "uri" : "imap:mail.example.com",
         "tls" : {
           "required" : true
         },
         "auth" : ["CRAM-MD5"]
       },

       {
         "name" : "Primary Mail",
         "service" : "pop3",
         "group" : "primary",
         "priority" : 1,
         "host" : "mail.example.com",
         "port" : 110,
         "tls" : {
           "required" : true
         },
         "auth" : ["CRAM-MD5"]
       },

       {
         "name" : "Internal Mail",
         "service" : "imap",
         "group" : "internal",
         "priority" : 2,
         "uri" : "imap:int.example.com",
         "tls" : {
           "required" : true
         },
         "auth" : ["CRAM-MD5"]
       },
```

```
       {
         "name" : "Internal Mail",
         "service" : "pop3",
         "group" : "internal",
         "priority" : 1,
         "host" : "int.example.com",
         "port" : 110,
         "tls" : {
           "required" : true
         },
         "auth" : ["CRAM-MD5"]
       }

       {
         "name" : "Internal Mail",
         "service" : "{example.com}webmail",
         "group" : "internal",
         "priority" : 1,
         "uri" : "https://int.example.com/webmail",
         "{example.com}bookmark": "https://int.example.com/webmail"
       }
     ]
   }
```

Appendix D. Example - multiple links

TODO: example of webfinger returning multiple "service-configuration" links.

Authors' Addresses

Andrew McMillan
Morphoss Ltd
6 Karoro Place
Porirua 5024
New Zealand

EMail: andrew@morphoss.com
URI: http://www.morphoss.com/

Cyrus Daboo
Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
USA

EMail: cyrus@daboo.name
URI: http://www.apple.com/