Network Working Group	M. Jones
Internet-Draft	Y. Goland
Intended status: Standards Track	Microsoft
Expires: June 15, 2012	December 13, 2011

Simple Web Discovery (SWD)
draft-jones-simple-web-discovery-02

Abstract

Simple Web Discovery (SWD) defines an HTTPS GET based mechanism to discover the location of a given type of service for a given principal starting only with a domain name.

Requirements Language

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** [RFC2119].

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

TOC

Simple Web Discovery (SWD) defines an HTTPS GET based mechanism to discover the location of a given type of service for a given principal starting only with a domain name. SWD requests use the x-www-form-urlencoded format to specify a URI for the principal and another URI for the type of service being sought. If the request is successful then the response, by default, is a JSON object containing an array of URIs that point to where the principal has instances of services of the requested type.

For example, let us say that a requester wants to discover where Joe keeps his calendar. The requester could take Joe's e-mail address, joe@example.com, and use its domain to create an HTTPS GET request of the following form (with long lines broken for display purposes only):

```
GET /.well-known/simple-web-discovery
    ?principal=mailto:joe@example.com
    &service=urn:example.org:service:calendar HTTP/1.1
Host: example.com

HTTP/1.1 200 OK
Content-Type: application/json

{
 "locations": ["http://calendars.example.net/calendars/joseph"]
}
```

Note: The request-URI is left unencoded in the above example for the sake of readability. The query parameters above would actually be encoded as `?principal=mailto%3Ajoe%40example.com&service=urn%3Aexample.org%3Aservice%3Acalendar`.

2. Simple Web Discovery Request

TOC

Domains that support SWD requests MUST make available a SWD server for their domain at the path `/.well-known/simple-web-discovery`. The syntax and semantics of `/.well-known` are defined in **RFC 5785** [RFC5785]. `simple-web-discovery` MUST point to a SWD server compliant with this specification.

SWD servers MUST support receiving SWD requests via TLS 1.2 as defined in **RFC 5246** [RFC5246] and MAY support other transport layer security mechanisms of equivalent security. SWD servers MUST reject SWD requests sent over plain HTTP or any other transport that does not provide both privacy and validation of the server's identity.

A SWD server is queried using an HTTPS GET request with the previously specified path along with a query segment containing an x-www-form-urlencoded form as defined in **HTML 4.01** [W3C.REC-html401-19991224]. The form MUST contain two name/value pairs that MUST appear exactly once, `principal` and `service`. Both name/value pairs MUST have values that are set to URIs (as defined in **RFC 3986** [RFC3986]). If any of the previous requirements are not met in a SWD request, then the request MUST be rejected with a 400 Bad Request.

The SWD request form MAY contain additional name/value pairs but if those name/value pairs are not recognized by the SWD server then the SWD server MUST ignore them for processing purposes.

The `principal` query component is a URI that identifies an entity. The `service` query component is a URI that identifies a service type. The semantics of the SWD query is "Please return the location(s) of instances of the specified service type associated with the specified principal". The definition of URIs used to identify principals and services are outside the scope of this specification.

3. Simple Web Discovery Responses

TOC

3.1. Response Containing One or More Locations

TOC

Unless another content-type is negotiated, a 200 OK response to a SWD request that contains the information requested MUST return content of type application/json as defined in **RFC 4627** [RFC4627]. The JSON response MUST contain a JSON object that contains a member pair whose name is the string `locations` and whose value is an array of strings that are each a URI pointing to a location where the desired service type belonging to the specified principal can be found. There are no semantics associated with the order in which the URIs are listed in the array.

The JSON object MAY contain other members but a receiver of the object MAY ignore any member pairs whose name it does not recognize.

3.2. Redirecting All Simple Web Discovery Requests

TOC

SWD requests by definition start off by being issued to the `/.well-known/simple-web-discovery` location. But locating a SWD server at a root location can prove inconvenient. To enable service level redirection, a SWD server MAY return a 200 OK to an HTTPS request with a content type of application/json (or whatever other content type has been negotiated) that contains a JSON object that contains a member pair whose name is the string `SWD_service_redirect` whose value is a JSON object with a member pair whose name is `location` and whose value is a string that encodes a URI. Optionally the JSON object value of `SWD_service_redirect` MAY also contain a member whose name is `expires` and whose value is a JSON number that encodes an integer.

A SWD compliant client MUST support the `SWD_service_redirect` response.

The JSON objects MAY contain other members but a receiver of the objects MAY ignore any pairs whose name it does not recognize.

The `location` member identifies the URI that the caller MUST redirect all SWD requests for that domain to until the `expires` time has passed. SWD requests for the redirected domain MUST be constructed by taking the URI returned in the `location` and using it as the base URI to which the SWD form arguments are then added as query parameters. The location URI MUST NOT include a query component.

The following is an example of redirect messages (with long lines broken for display purposes only):

```
GET /.well-known/simple-web-discovery
    ?principal=mailto:joe@example.com
    &service=urn:example.org:service:calendar HTTP/1.1
Host: example.com

HTTP/1.1 200 OK
Content-Type: application/json

{
 "SWD_service_redirect":
  {
   "location": "https://swd.example.com/swd_server",
   "expires": 1300752001
  }
}

GET /swd_server
    ?principal=mailto:joe@example.com
    &service=urn:example.org:service:calendar HTTP/1.1
Host: swd.example.com
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
 "locations": ["http://calendars.example.net/calendars/joseph"]
}
```

Note: The request-URIs are left unencoded in the above example for the sake of readability.

The `location` URI MUST be an HTTPS URL.

The optional `expires` member identifies the point in time at which the caller MUST NOT redirect its SWD requests for that domain to the previously obtained `location` and MUST instead return to the `/.well-known/simple-web-discovery` location. The value of the `expires` member MUST encode the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the desired date/time. See **RFC 3339** [RFC3339] for details regarding date/times in general and UTC in particular. If the `expires` value is in the past or if the value is more than one hour in the future then the response MUST be treated as if it didn't contain an `expires` value.

If the `expires` value is omitted or if its value is incorrect then the `expires` value MUST be treated as having a value of exactly one hour into the future.

If a JSON response is received that contains both a member pair with the name `SWD_service_redirect` and a member pair with the name `locations` as children of the object root then the `SWD_service_redirect` member pair MUST be ignored.

3.3. 401 Unauthorized Response

A SWD server MAY respond to a request with a 401 Unauthorized Response, as described in **RFC 2616** [RFC2616], Section 10. Per the RFC, the request MAY be repeated with a suitable Authorization header field. Authorization information may be communicated in this manner, including a JSON Web Token **[JWT]**.

3.4. Other HTTP 1.1 Responses

A SWD server MAY return other HTTP 1.1 responses, including 404 Not Found, 400 Bad Request, and 403 Forbidden. SWD implementations MUST correctly handle these responses.

4. IANA Considerations

Per **RFC 5785** [RFC5785], the following registration template is offered:

URI suffix	simple-web-discovery
Change controller	IETF
Specification document	This RFC

5. Security Considerations

SWD responses can contain confidential information. Therefore a, general approach is used to require TLS in all cases. But TLS can only provide for privacy and server validation, it cannot validate that the requester is authorized to see the results of a query. The exact mechanism used to determine if the requester is authorized to see the result of the query is outside the scope of this specification.

Because SWD responses can contain confidential information, the requestor may need authorization to receive them. Standard HTTP authorization mechanisms MAY be employed to request authorized access, including the use of an HTTP Authorization header field in requests, which in turn, may contain a JSON Web Token [JWT], among other authorization data formats.

The ability to redirect an entire SWD server as defined in this document is an obvious attack point. This is another reason why we have mandated TLS, so as to be sure that the redirect can only be received over a secure connection. We have also put in the upper limit of 60 minutes for a redirect so as to provide a path for regaining control over queries should a successful attack be launched to return false redirects.

The `SWD_service_redirect` capability may cause unanticipated failures in cases where a requestor may have permissions to discover content at the original SWD endpoint but not the one redirected to, or vice-versa.

6. References

TOC

6.1. Normative References

TOC

[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[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 (TXT, PS, PDF, HTML, XML).
[RFC3339]	Klyne, G., Ed. and C. Newman, “Date and Time on the Internet: Timestamps,” RFC 3339, July 2002 (TXT, HTML, XML).
[RFC3986]	Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” STD 66, RFC 3986, January 2005 (TXT, HTML, XML).
[RFC4627]	Crockford, D., “The application/json Media Type for JavaScript Object Notation (JSON),” RFC 4627, July 2006 (TXT).
[RFC5246]	Dierks, T. and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” RFC 5246, August 2008 (TXT).
[RFC5785]	Nottingham, M. and E. Hammer-Lahav, “Defining Well-Known Uniform Resource Identifiers (URIs),” RFC 5785, April 2010 (TXT).
[W3C.REC-html401-19991224]	Jacobs, I., Raggett, D., and A. Hors, “HTML 4.01 Specification,” World Wide Web Consortium Recommendation REC-html401-19991224, December 1999 (HTML).

6.2. Informative References

TOC

[JWT] Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Token (JWT),” December 2011.

Appendix A. Document History

TOC

-02

- Update examples to use example.{com,net,org} domain names.
- Provide encoded representation of the request-URI query parameters for the first example request.
- Changed "200 O.K." to "200 OK".
- Respect line length restrictions in examples.
- No normative changes.

-01

- Refresh draft before expiration of -00. No normative changes.

-00

- Initial version.

TOC

Authors' Addresses

Michael B. Jones
Microsoft
Email: mbj@microsoft.com
URI: http://self-issued.info/

Yaron Y. Goland
Microsoft
Email: yarong@microsoft.com