Simple Service Discovery Protocol/1.0
<draft-cai-ssdp-v1-01.txt>

Status of this Memo

   This document is an Internet-Draft and is in full conformance with
   all provisions of Section 10 of RFC2026.

   This document is an Internet-Draft. Internet-Drafts are working
   documents of the Internet Engineering Task Force (IETF), its areas,
   and its working groups. Note that other groups may also distribute
   working documents as Internet-Drafts.

   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 material or to cite them other than as "work in progress."

   The list of current Internet-Drafts can be accessed at
   http://www.ietf.org/ietf/1id-abstracts.txt.

   The list of Internet-Draft Shadow Directories can be accessed at
   http://www.ietf.org/shadow.html.

Abstract

   The Simple Service Discovery Protocol (SSDP) provides a mechanism
   where by network clients, with little or no static configuration,
   can discover desired network services. SSDP uses HTTP over multicast
   and unicast UDP to provide two functions: OPTIONS and ANNOUNCE.
   OPTIONS is used to determine if a desired network service exists on
   the network. ANNOUNCE is used by network services to announce their
   existence.

1.   Introduction

With the growing number of small peer-to-peer TCP/IP networks, such
as home or small office networks, computer users need a way to
discover resources in a network easily, quickly, dynamically, and
without any a priori knowledge. This document proposes a discovery
protocol that meets this user requirement.

The proposed protocol is called the Simple Service Discovery
Protocol (or SSDP). SSDP performs only discovery. It leaves any
additional service description and/or negotiation to a higher layer
service-specific protocol.

Using SSDP, operating systems and applications can discover SSDP-
enabled services in a network dynamically by issuing service
queries. Services that match the queries respond with their network
locations and protocols to be used to communicate with them. In
addition, services can make their presence known through unsolicited
announcements.

2.   Terminology

Service Provider
       A resource that provides one or more network services, such
       as printing.

SSDP Server
       A resource that advertises services and responds to service
       queries on behalf of one or more Service Providers.

SSDP Client
       A resource that locates services of interest by sending out
       service queries and gathering responses.

In addition, 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 [2].

3.   Protocol Overview

SSDP-enabled services are discoverable in two ways. Firstly, a SSDP
Client sends a service query using an OPTIONS message. The OPTIONS
message is encapsulated in an UDP packet and sent to a reserved
multicast IP address and port. SSDP Servers MUST always listen for
service queries. They MUST respond to the SSDP Client's query using
one or more Reply messages, if they represent Service Providers that
match those specified in the query. In addition, SSDP Servers MUST
wait for a randomly selected period between MIN_REPLY_DELAY_INTERVAL

and MAX_REPLY_DELAY_INTERVAL (inclusively) before it sends any Reply
message.

The SSDP Client times out if it does not receive a Reply message for
a service query within RETRY_TIMEOUT_INTERVAL. The SSDP Client
SHOULD retransmit another OPTIONS message at a randomly selected
time between MIN_RETRY_INTERVAL and MAX_RETRY_INTERVAL (inclusively)
after the timeout. It SHOULD re-send the OPTIONS message up to
MAX_RETRIES times. Re-sent OPTIONS messages SHOULD use the same
Request-ID as the original request.

4.   Multicast and UDP

SSDP transmits its OPTIONS and ANNOUNCE messages using multicast
UDP. A multicast IP address and two port numbers will be reserved
for the SSDP. SSDP Servers MUST listen on the reserved address and
the first port for OPTIONS messages. They MUST send ANNOUNCE
messages to the reserved address and the second port.

[Ed. Note: It is the intention of the authors to produce a separate
draft that specifies how to use HTTP over unicast and multicast UDP.
The SSDP draft will then reference this other draft.]

5.   Header Definitions

SSDP uses the HTTP message syntax, methods and headers. Small
changes have been made in order to enable proper functionality over
UDP multicast/unicast. HTTP headers and methods that appear in this
document but are not defined explicitly MUST follow their
definitions in the HTTP specification [3].

5.1. Host Header

The Host header in HTTP specifies the Internet host and port number
of the resource being requested. In a multicast environment the
value of the Host header should be the multicast address and port.

[Ed. Note: Do we really need to use the host head at all in the case
of multicast requests? If so, why? The only argument I can come up
with is that you may want to only address a subset of the multicast
recipients. But this would require that the host header be able to
take multiple values and probably include URIs as well as domain
names and IP addresses. This seems like a slippery slope.]

5.2. Alt-Locations Header

The Alt-Locations header is used in conjunction with the Location

header. When present, it lists alternative location(s) to the one
specified in the Location header for completion of the request.

Alt-Locations = "Alt-Locations" ":" 1*(_<_ AbsoluteURI _>_)

AbsoluteURI is defined in the HTTP specification [3].

The Alt-Locations header is an extension of the Location header
although it MAY be used without the Location header. Resources
SHOULD first try to resolve the URI in the Location header and only
continue on to the Alt-Locations header if the URI in the Location
header could not be successfully resolved or if there is no Location
header. Entries in the Alt-Locations header should then be resolved,
in order.

[Ed. Note: Since there are at least two specs which want to use this
header, it is likely this header will be split out into its own
spec.]
[Ed. Note: Do we need an implementation note telling people to
randomize the order of the entries in the alt-locations header if
they want a good spread?]

5.3. Request-ID Header

The Request-ID header is used to correlate responses from different
hosts for a particular request and to distinguish responses for
different service requests.

Request-ID = "Request-ID" ":" AbsoluteURI

In order to prevent the possibility of collisions Request-Ids must
be unique across all requests from all resources for all time. One
URI able to meet this requirement is the UUID URI that is defined as
follows:

UUID_URI = _UUID_ _:_ UUID [Extension]

The UUID and Extension productions are defined in [4].

[Ed. Note: It is the author's intention to put the UUID URI
definition in its own draft and to then make SSDP dependent on that
draft.]

[Ed. Note: Technically a Request-ID only needs to be unique to the
resource that sent it. However this requires that the Request-ID be
associated with some other value in order to tell one Request-ID
from another. The most obvious value would be the IP address that
sent the multicast UDP request. However IP addresses are constantly

re-assigned and so using them to differentiate Request-Ids could
cause a collision.]

6.   Method Definitions

6.1. OPTIONS

6.1.1.   Request

SSDP uses the OPTIONS method to determine if services are available
on the network.

The Request-URI is the URI of the service to be discovered. The
HTTP/1.1 reserved Request-URI _*_ is interpreted as meaning all
services.

[Ed. Note: Need to add a section explaining how URIs are used to
identify services.]

The OPTIONS message MUST include a Host header and a Request-ID
header. Optionally, it MAY contain a Content-Length header, a
Content-Type header and an Entity Body [3]. The Entity Body is used
to provide additional information to select a Service Provider. If
the resource processing the request does not understand the request
body then it MUST ignore the request-body.

Below is an example of an OPTIONS message.

OPTIONS /ietf/ipp/printer HTTP/1.1
Host: 239.255.255.254:999
Request-ID: uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
Alt-Locations: <httpu://foo.com/bar:1270><http://foo.com/bar:80>

Please note that the multicast address and port are not the SSDP
reserved multicast address and port and are only included as an
example.

In this example the sender requests that a response be sent using
HTTP unicast UDP if at all possible. If not then the response should
be sent using normal HTTP. Note, however, that this requires
establishing a TCP/IP connection and then sending a response. This
behavior is not provided for by [3]. However, by including an HTTP
URL in the Location header of an HTTPU or HTTPMU request the sender
is indicating their support for receiving the response directly over
TCP/IP.

6.1.2.   Response

If a Location and/or Alt-Locations header is included in the request
then the response MUST be directed to the first URI the resource has
the ability to respond to. If the resource can not respond to any of
the listed URIs then a response MUST NOT be sent.

If no Location and/or Alt-Locations header is included then the
resource MUST respond to the sender using unicast UDP.

In either case the response MUST contain the same Request-ID as the
request.

The URI scheme name httpu is reserved for use with HTTP unicast UDP.
The URI scheme name httpmu is reserved for use with HTTP multicast
UDP.

[Ed. Note: Both schemes will be placed in the HTTP over UDP
document.]

All resources that are able to accept incoming multicast UDP HTTP
requests MUST be able to respond using unicast UDP HTTP.

A 200 response to a multicast UDP OPTIONS request indicates that the
response includes all OPTIONS information. A 3xx series response may

also be used to direct the client to another location where they
SHOULD retry their request.

The Reply message SHOULD include a Location and/or Alt-Locations
headers. Optionally it MAY contain a Content-Length header, a
Content-Type header and an Entity Body. If the resource does not
understand the response body then the default behavior is to ignore
the response body.

Below is an example of a Reply message.

HTTP/1.1 302 Found
Request-ID: uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
Location: ipp://printServer1/printerA
Alt-Locations: <lpr://printServer1/printerA>

6.2. ANNOUNCE

6.2.1.   Request

The ANNOUNCE method represents an unsolicited message from an SSDP
Server to announce the availability of Service Providers. The
Request-URI is the URI of the service being announced.

The Location header [3] specifies the protocol end-point of a
Service Provider. This header MUST contain the protocol that can be
used to communicate with the Service Provider.

Furthermore, if the Service Provider supports multiple protocols, an
Alt-Location header is used to identify the additional protocol end-
points.

The ANNOUNCE message SHOULD include a Location header. It SHOULD
also include an Alt-Locations header if the Service Provider has
more than one URI. The ANNOUNCE method SHOULD NOT contain the
Request-ID header. Optionally, it MAY contain a Content-Length
header, a Content-Type header and an Entity Body. If the resource
processing the request does not understand the request body then it
MUST ignore the request-body.

Below is an example of an ANNOUNCE message.

```
ANNOUNCE dmtf:printer HTTP/1.1
Host: 239.255.255.254:999
Location: ipp://printServer1/printerA
Alt-Locations: <lpr://printServer1/printerA>
```

Please note that the multicast address and port are not the SSDP
reserved multicast address and port and are only included as an
example.

[Ed. Note: Using the service class as the request-URI of the
announce method certainly seems nifty, but I'm not sure it is the
right choice. From a practical point of view there appear to be two
choices. Set the Request-URI for all ANNOUNCE methods to be "*" or
use some sort of URI. "*" isn't terribly useful as it means that all
ANNOUNCE methods get processed by one and only one resource. However
using the service class as the request-URI means that we end up with
two very different types of resources sharing the same request-URI.
One type is the type who takes in and responds to OPTIONS methods.
The second type is the type who takes in and records ANNOUNCE
methods. The trick is that the relationship between these two
resources is circular. The resource which responds to OPTIONS
methods is the one sending the ANNOUNCE methods. The one who records
ANNOUNCE methods is the one who sends OPTIONS methods. But they both
have the same request-URI. I'm not sure if this is really a problem
but it is food for thought.]

6.2.2.   Response

   There is no response to an ANNOUNCE request.

7.   Practical Considerations

7.1. Multiple-Functional Devices

   A multiple-functional device provides several services in the same
   unit. The SSDP Server for a multiple-functional device MUST send an
   ANNOUNCE message for each of the device functions (i.e., Service
   Providers). For example, a device that combines a scanner and a fax
   machine would send out the following ANNOUNCE messages:

       Announce 1:
   ANNOUNCE /ietf/scanner HTTP/1.1
   HOST: 239.255.255.254:999
   Location: scannerProtocol://deviceY

       Announce 2:
   ANNOUNCE itu://devices/fax HTTP/1.1
   HOST: 239.255.255.254:999
   Location: faxProtocol://deviceY

7.2. SSDP Server Proxy

   For legacy devices that do not have their own SSDP Servers, a SSDP
   Server Proxy can be configured to announce services and respond to
   service queries on behalf of legacy devices. With a SSDP Server
   Proxy, legacy devices can be made discoverable in the same way as
   devices with built-in SSDP Servers.

8.   Security Considerations

   SSDP does not provide a mechanism to sign or encrypt its requests.
   The authors feel that authentication/encryption for HTTP UDP is of
   sufficient general interest that it should be addressed in its own
   specification. SSDP will be made dependent upon this
   authentication/encryption specification.

   [Ed. Note: Section 6.19 of the SIP spec
   (http://www.ietf.org/internet-drafts/draft-ietf-mmusic-sip-12.txt)
   provides an example of how one can encrypt individual HTTP
   messages.]

9.   Appendix A: Constants

   MAX_REPLY_DELAY_INTERVAL
         A parameter that indicates the maximum number of seconds that
         a SSDP Server MUST wait before it responds to a OPTIONS
         message. This parameter SHOULD be configurable and the
         default value SHOULD be 8.

MIN_REPLY_DELAY_INTERVAL
        A parameter that indicates the minimum number of seconds that
        a SSDP Server MUST wait before it responds to a OPTIONS
        message. This parameter SHOULD be configurable and the
        default value SHOULD be 0.

MAX_RETRIES
        A parameter that indicates the maximum number of times that a
        SSDP Client SHOULD repeat sending OPTIONS messages for a
        particular service query. This parameter SHOULD be
        configurable and the default value SHOULD be 3.

MAX_RETRY_INTERVAL
        A parameter that indicates the maximum number of seconds that
        a SSDP Client MUST wait (after each timeout) before it
        retransmits another OPTIONS message for a particular service
        query. This parameter SHOULD be configurable and the default
        value SHOULD be 10.

MIN_RETRY_INTERVAL
        A parameter that indicates the minimum number of seconds that
        a SSDP Client MUST wait (after each timeout) before it
        retransmits another OPTIONS message for a particular service
        query. This parameter SHOULD be configurable and the default
        value SHOULD be 3.

RETRY_TIMEOUT_INTERVAL
        A parameter that indicates the number of seconds that a SSDP
        Client MUST wait for a Reply message to a Discovery message
        before it times out. This parameter SHOULD be configurable
        and the default value SHOULD be 15.

10. Acknowledgements

This document is a joint contribution of numerous people, including
Ting Cai, Paul Leach, Munil Shah, Holly Knight, Ye Gu, Peter Ford,
Pradeep Bahl, and Shivaun Albright. Special thanks to Holly Knight
for her patient reviews and detailed comments. We also appreciate
valuable feedback from Paul Moore, Babak Jahromi, Brandon Watson,
Michel Guittet, Todd Fisher, and Craig White.

11. References

1.   T. Berners-Lee, R. Fielding and L. Masinter. Uniform Resource
     Identifiers (URI): Generic Syntax. RFC 2396, August 1998.
2.   S. Bradner. Key words for use in RFCs to Indicate Requirement

           Levels. RFC 2119, March 1997.
   3.   R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-
         Lee. Hypertext Transfer Protocol -- HTTP/1.1. RFC 2068,
         January 1997.
   4.   Y. Goland, E. Whitehead, A. Faizi, S. Carter, and D. Jensen.
         HTTP Extensions for Distributed Authoring _ WEBDAV. RFC 2518,
         February 1999.

12.   Author's Addresses

      Ting Cai, Paul Leach, Ye Gu, Yaron Y. Goland
      Microsoft Corporation
      One Microsoft Way
      Redmond, WA 98052

      Email: {tingcai, paulle, yegu, yarong}@microsoft.com

      Shivaun Albright
      Hewlett-Packard Company
      Roseville, CA

      Email: SHIVAUN_ALBRIGHT@HP-Roseville-om2.om.hp.com

      This document will expire in September 1999.