

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 26, 2021

T. Lemon  
S. Cheshire  
Apple Inc.  
February 22, 2021

Multicast DNS Discovery Relay  
draft-ietf-dnssd-mdns-relay-04

## Abstract

This document complements the specification of the Discovery Proxy for Multicast DNS-Based Service Discovery. It describes a lightweight relay mechanism, a Discovery Relay, which, when present on a link, allows remote clients, not attached to that link, to perform mDNS discovery operations on that link.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 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 material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 26, 2021.

## Copyright Notice

Copyright (c) 2021 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.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. Protocol Overview . . . . .	6
3.1. Connections between Clients and Relays (overview) . . . . .	6
3.2. mDNS Messages On Multicast Links . . . . .	7
4. Connections between Clients and Relays (details) . . . . .	8
5. Traffic from Relays to Clients . . . . .	10
6. Traffic from Clients to Relays . . . . .	12
7. Discovery Proxy Behavior . . . . .	13
8. DSO TLVs . . . . .	14
8.1. mDNS Link Data Request . . . . .	14
8.2. mDNS Link Data Discontinue . . . . .	14
8.3. Link Identifier . . . . .	15
8.4. Encapsulated mDNS Message . . . . .	15
8.5. IP Source . . . . .	15
8.6. Link State Request . . . . .	16
8.7. Link State Discontinue . . . . .	16
8.8. Link Available . . . . .	16
8.9. Link Unavailable . . . . .	16
8.10. Link Prefix . . . . .	17
9. Provisioning . . . . .	18
9.1. Provisioned Objects . . . . .	19
9.1.1. Multicast Link . . . . .	20
9.1.2. Discovery Proxy . . . . .	21
9.1.3. Discovery Relay . . . . .	22
9.2. Configuration Files . . . . .	23
9.3. Discovery Proxy Private Configuration . . . . .	25
9.4. Discovery Relay Private Configuration . . . . .	25
10. Security Considerations . . . . .	26
11. IANA Considerations . . . . .	27
12. Acknowledgments . . . . .	27
13. References . . . . .	28
13.1. Normative References . . . . .	28
13.2. Informative References . . . . .	29
Authors' Addresses . . . . .	30

## 1. Introduction

This document defines a Discovery Relay. A Discovery Relay is a companion technology that works in conjunction with Discovery Proxies, and other clients.

The Discovery Proxy for Multicast DNS-Based Service Discovery [RFC8766] is a mechanism for discovering services on a subnetted network through the use of Discovery Proxies. Discovery Proxies issue Multicast DNS (mDNS) requests [RFC6762] on various multicast links in the network on behalf of a remote host performing DNS-Based Service Discovery [RFC6763].

In the original Discovery Proxy specification, it was imagined that for every multicast link on which services will be discovered, a host will be present running a full Discovery Proxy. This document introduces a lightweight Discovery Relay that can be used in conjunction with a central Discovery Proxy to provide discovery services on a multicast link without requiring a full Discovery Proxy on every multicast link.

The primary purpose of a Discovery Relay is providing remote virtual interface functionality to Discovery Proxies, and this document is written with that usage in mind. However, in principle, a Discovery Relay could be used by any properly authorized client. In the context of this specification, a Discovery Proxy is a client to the Discovery Relay. This document uses the terms "Discovery Proxy" and "Client" somewhat interchangeably; the term "Client" is used when we are talking about the communication between the Client and the Relay, and the term "Discovery Proxy" when we are referring specifically to a Discovery Relay Client that also happens to be a Discovery Proxy. One example of another kind of device that can be a client of a Discovery Relay is an Advertising Proxy [AdProx].

The Discovery Relay operates by listening for TCP connections from Clients. When a Client connects, the connection is authenticated and secured using TLS. The Client can then specify one or more multicast links from which it wishes to receive mDNS traffic. The Client can also send messages to be transmitted on its behalf on one or more of those multicast links. DNS Stateful Operations (DSO) [RFC8490] is used as a framework for conveying interface and IP header information associated with each message. DSO formats its messages using type-length-value (TLV) data structures. This document defines additional DSO TLV types, used to implement the Discovery Relay functionality.

The Discovery Relay functions essentially as a set of one or more remote virtual interfaces for the Client, one on each multicast link to which the Discovery Relay is connected. In a complex network, it

is possible that more than one Discovery Relay will be connected to the same multicast link; in this case, the Client ideally should only be using one such Relay Proxy per multicast link, since using more than one will generate duplicate traffic.

How such duplication is detected and avoided is out of scope for this document; in principle it could be detected using HNCP [RFC7788] or configured using some sort of orchestration software in conjunction with NETCONF [RFC6241] or CPE WAN Management Protocol [TR-069].

Use of a Discovery Relay can be considered similar to using Virtual LAN (VLAN) trunk ports to give a Discovery Proxy device a virtual presence on multiple links or broadcast domains. The difference is that while a VLAN trunk port operates at the link layer and delivers all link-layer traffic to the Discovery Proxy device, a Discovery Relay operates further up the network stack and selectively delivers only relevant Multicast DNS traffic. Also, VLAN trunk ports are generally only available within a single administrative domain and require link-layer configuration and connectivity, whereas the Discovery Relay protocol, which runs over TCP, can be used between any two devices with IP connectivity to each other.

## 2. Terminology

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 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

The following definitions may be of use:

**Client** A network service that uses a Discovery Relay to send and receive mDNS multicast traffic on a remote link, to enable it to communicate with mDNS Agents on that remote link.

**mDNS Agent** A host which sends and/or responds to mDNS queries directly on its local link(s). Examples include network cameras, networked printers, networked home electronics, etc.

**Discovery Proxy** A network service which receives well-formed questions using the DNS protocol, performs multicast DNS queries to find answers to those questions, and responds with those answers using the DNS protocol. A Discovery Proxy that can communicate with remote mDNS Agents, using the services of a Discovery Relay, is a Client of the Discovery Relay.

**Discovery Relay** A network service which relays mDNS messages received on a local link to a Client, and on behalf of that Client can transmit mDNS messages on a local link.

**multicast link** A maximal set of network connection points, such that any host connected to any connection point in the set may send a packet with a link-local multicast destination address (specifically the mDNS link-local multicast destination address [RFC6762]) that will be received by all hosts connected to all other connection points in the set. Note that it is becoming increasingly common for a multicast link to be smaller than its corresponding unicast link. For example it is becoming common to have multiple Wi-Fi access points on a shared Ethernet backbone, where the multiple Wi-Fi access points and their shared Ethernet backbone form a single unicast link (a single IPv4 subnet, or single IPv6 prefix) but not a single multicast link. Unicast packets sent directly between two hosts on that IPv4 subnet or IPv6 prefix, without passing through an intervening IP-layer router, are correctly delivered, but multicast packets are not forwarded between the various Wi-Fi access points. Given the slowness of Wi-Fi multicast [I-D.ietf-mboned-ieee802-mcast-problems], having a packet that may be of interest to only one or two end systems transmitted to hundreds of devices, across multiple Wi-Fi access points, is especially wasteful. Hence the common configuration decision to not forward multicast packets between Wi-Fi access points is very reasonable. This further motivates the need for technologies like Discovery Proxy and Discovery Relay to facilitate discovery on these networks.

**allow-list** A list of one or more IP addresses from which a Discovery Relay may accept connections.

**silently discard** When a message that is not supported or not permitted is received, and the required response to that message is to "silently discard" it, that means that no response is sent by the service that is discarding the message to the service that sent it. The service receiving the message may log the event, and may also count such events: "silently" does not preclude such behavior.

Take care when reading this document not to confuse the terms "Discovery Proxy" and "Discovery Relay". A Discovery Proxy [RFC8766] provides Multicast DNS discovery service to remote clients. A Discovery Relay is a simple software entity that provides virtual link connectivity to one or more Discovery Proxies or other Discovery Relay clients.

### 3. Protocol Overview

This document describes a way for a Client to communicate with mDNS agents on remote multicast links to which the client is not directly connected, using a Discovery Relay. As such, there are two parts to the protocol: connections between Clients and Discovery Relays, and communications between Discovery Relays and mDNS agents.

#### 3.1. Connections between Clients and Relays (overview)

Discovery Relays listen for incoming connection requests. Connections between Clients and Discovery Relays are established by Clients. Connections are authenticated and encrypted using TLS, with both client and server certificates. Connections are long-lived: a Client is expected to send many queries over a single connection, and Discovery Relays will forward all mDNS traffic from subscribed interfaces over the connection.

The stream encapsulated in TLS will carry DNS frames as in the DNS TCP protocol [RFC1035] Section 4.2.2. However, all messages will be DSO messages [RFC8490]. There will be four types of such messages between Discovery Relays and Clients:

- o Control messages from Client to Relay
- o Link status messages from Relay to Client
- o Encapsulated mDNS messages from Client to Relay
- o Encapsulated mDNS messages from Relay to Client

Clients can send four different control messages to Relays: Link State Request, Link State Discontinue, Link Data Request and Link Data Discontinue. The first two are used by the Client to request that the Relay report on the set of links that can be requested, and to request that it discontinue such reporting. The second two are used by the Client to indicate to the Discovery Relay that mDNS messages from one or more specified multicast links are to be relayed to the Client, and to subsequently stop such relaying.

Link Status messages from a Discovery Relay to the Client inform the Client that a link has become available, or that a formerly-available link is no longer available.

Encapsulated mDNS messages from a Discovery Relay to a Client are sent whenever an mDNS message is received on a multicast link to which the Discovery Relay has subscribed.

Encapsulated mDNS messages from a Client to a Discovery Relay cause the Discovery Relay to transmit the mDNS message on the specified multicast link to which the Discovery Relay host is directly attached.

During periods with no traffic flowing, Clients are responsible for generating any necessary keepalive traffic, as stated in the DSO specification [RFC8490].

### 3.2. mDNS Messages On Multicast Links

Discovery Relays listen for mDNS traffic on all configured multicast links that have at least one active subscription from a Client. When an mDNS message is received on a multicast link, it is forwarded on every open Client connection that is subscribed to mDNS traffic on that multicast link. In the event of congestion, where a particular Client connection has no buffer space for an mDNS message that would otherwise be forwarded to it, the mDNS message is not forwarded to it. Normal mDNS retry behavior is used to recover from this sort of packet loss. Discovery Relays are not expected to buffer more than a few mDNS packets. Excess mDNS packets are silently discarded. In practice this is not expected to be a issue. Particularly on networks like Wi-Fi, multicast packets are transmitted at rates ten or even a hundred times slower than unicast packets. This means that even at peak multicast packets rates, it is likely that a unicast TCP connection will be able to carry those packets with ease.

Clients send encapsulated mDNS messages they wish to have sent on their behalf on remote multicast link(s) on which the Client has an active subscription. A Discovery Relay will not transmit mDNS packets on any multicast link on which the Client does not have an active subscription, since it makes no sense for a Client to ask to have a query sent on its behalf if it's not able to receive the responses to that query.

#### 4. Connections between Clients and Relays (details)

When a Discovery Relay starts, it opens a passive TCP listener to receive incoming connection requests from Clients. This listener may be bound to one or more source IP addresses, or to the wildcard address, depending on the implementation. When a connection is received, the relay must first validate that it is a connection to an IP address to which connections are allowed. For example, it may be that only connections to ULAs are allowed, or to the IP addresses configured on certain interfaces. If the listener is bound to a specific IP address, this check is unnecessary.

If the relay is using an IP address allow-list, the next step is for the relay to verify that the source IP address of the connection is on its allow-list. If the connection is not permitted either because of the source address or the destination address, the Discovery Relay closes the connection. If possible, before closing the connection, the Discovery Relay first sends a TLS `user_canceled` alert ([RFC8446] Section 6.1). Discovery Relays SHOULD refuse to accept TCP connections to invalid destination addresses, rather than accepting and then closing the connection, if this is possible.

Otherwise, the Discovery Relay will attempt to complete a TLS handshake with the Client. Clients are required to send the `post_handshake_auth` extension ([RFC8446] Section 4.2.5). If a Discovery Relay receives a `ClientHello` message with no `post_handshake_auth` extension, the Discovery Relay rejects the connection with a `certificate_required` alert ([RFC8446] Section 6.2).

Once the TLS handshake is complete, the Discovery Relay MUST request post-handshake authentication ([RFC8446] Section 4.6.2). If the Client refuses to send a certificate, or the key presented does not match the key associated with the IP address from which the connection originated, or the `CertificateVerify` does not validate, the connection is dropped with the `TLS access_denied` alert ([RFC8446] Section 6.2).

Clients MUST validate server certificates. If the client is configured with a server IP address and certificate, it can validate the server by comparing the certificate offered by the server to the certificate that was provided: they should be the same. If the certificate includes a Distinguished Name that is a fully-qualified domain name, the client SHOULD present that domain name to the server in an SNI request.

Rather than being configured with an IP address and a certificate, the client may be configured with the server's FQDN. In this case, the client uses the server's FQDN as a Authentication Domain Name



[RFC8310] Section 7.1, and uses the authentication method described in [RFC8310] section 8.1, if the certificate is signed by a root authority the client trusts, or the method described in section 8.2 of the same document if not. If neither method is available, then a locally-configured copy of the server certificate can be used, as in the previous paragraph.

Once the connection is established and authenticated, it is treated as a DNS TCP connection [RFC7766].

Aliveness of connections between Clients and Relays is maintained as described in Section 4 of the DSO specification [RFC8490]. Clients must also honor the 'Retry Delay' TLV (section 5 of [RFC8490]) if sent by the Discovery Relay.

Clients SHOULD avoid establishing more than one connection to a specific Discovery Relay. However, there may be situations where multiple connections to the same Discovery Relay are unavoidable, so Discovery Relays MUST be willing to accept multiple connections from the same Client.

In order to know what links to request, the Client can be configured with a list of links supported by the Relay. However, in some networking contexts, dynamic changes in the availability of links are likely; therefore Clients may also use the Report Link Changes TLV to request that the Relay report on the availability of its links. In some contexts, for example when debugging, a Client may operate with no information about the set of links supported by a relay, simply relying on the relay to provide one.

## 5. Traffic from Relays to Clients

The mere act of connecting to a Discovery Relay does not result in any mDNS traffic being forwarded. In order to request that mDNS traffic from a particular multicast link be forwarded on a particular connection, the Client must send one or more DSO messages, each containing a single mDNS Link Data Request TLV (Section 8.1) indicating the multicast link from which traffic is requested.

When an mDNS Link Data Request message is received, the Discovery Relay validates that it recognizes the link identifier, and that forwarding is enabled for that link. If both checks are successful, it MUST send a response with RCODE=0 (NOERROR). If the link identifier is not recognized, it sends a response with RCODE=3 (NXDOMAIN/Name Error). If forwarding from that link to the Client is not enabled, it sends a response with RCODE=5 (REFUSED). If the relay cannot satisfy the request for some other reason, for example resource exhaustion, it sends a response with RCODE=2 (SERVFAIL).

If the requested link is valid, the Relay begins forwarding all mDNS messages from that link to the Client. Delivery is not guaranteed: if there is no buffer space, packets will be dropped. It is expected that regular mDNS retry processing will take care of retransmission of lost packets. The amount of buffer space is implementation dependent, but generally should not be more than the bandwidth delay product of the TCP connection [RFC7323]. The Discovery Relay should use the TCP\_NOTSENT\_LOWAT mechanism [NOTSENT][PRIO] or equivalent, to avoid building up a backlog of data in excess of the amount necessary to have in flight to fill the bandwidth delay product of the TCP connection.

Encapsulated mDNS messages from Relays to Clients are framed within DSO messages. Each DSO message can contain multiple TLVs, but only a single encapsulated mDNS message is conveyed per DSO message. Each forwarded mDNS message is sent in an Encapsulated mDNS Message TLV (Section 8.4). The source IP address and port of the message MUST be encoded in an IP Source TLV (Section 8.5). The multicast link on which the message was received MUST be encoded in a Link Identifier TLV (Section 8.3). As described in the DSO specification [RFC8490], a Client MUST silently ignore unrecognized Additional TLVs in mDNS messages, and MUST NOT discard mDNS messages that include unrecognized Additional TLVs.

A Client may discontinue listening for mDNS messages on a particular multicast link by sending a DSO message containing an mDNS Link Data Discontinue TLV (Section 8.2). The Discovery Relay MUST discontinue forwarding mDNS messages when the Link Data Discontinue request is received. However, messages from that link that had previously been

queued may arrive after the Client has discontinued its listening. The Client should silently discard such messages. The Discovery Relay does not respond to the Link Data Discontinue message other than to discontinue forwarding mDNS messages from the specified links.

## 6. Traffic from Clients to Relays

Like mDNS traffic from relays, each mDNS message sent by a Client to a Discovery Relay is communicated in an Encapsulated mDNS Message TLV (Section 8.4) within a DSO message. Each message MUST contain exactly one Link Identifier TLV (Section 8.3). The Discovery Relay will transmit the mDNS message to the mDNS port and multicast address on the link specified in the message using the specified IP address family.

Although the communication between Clients and Relays uses the DNS stream protocol and DNS Stateless Operations, there is no case in which a Client would legitimately send a DNS query (or anything else other than a DSO message) to a Relay. Therefore, if a Relay receives any message other than a DSO message, it MUST immediately abort that DSO session with a TCP reset (RST).

When defining this behavior, the working group considered making it possible to specify more than one link identifier in an mDNSMessage TLV. A superficial evaluation of this suggested that this might be a useful optimization, since when a query is issued, it will often be issued to all links. However, on many link types, like Wi-Fi, multicast traffic is expensive [I-D.ietf-mboned-ieee802-mcast-problems] and should be generated frugally, so providing convenient ways to generate additional multicast traffic was determined to be an unwise optimization. In addition, because of the way mDNS handles retries, it will almost never be the case that the exact same message will be sent on more than one link. Therefore, the complexity that this optimization adds is not justified by the potential benefit, and this idea has been abandoned.

## 7. Discovery Proxy Behavior

Discovery Proxies treat multicast links for which Discovery Relay service is being used as if they were virtual interfaces; in other words, a Discovery Proxy serving multiple remote multicast links using multiple remote Discovery Relays behaves the same as a Discovery Proxy serving multiple local multicast links using multiple local physical network interfaces. In this section we refer to multicast links served directly by the Discovery Proxy as locally-connected links, and multicast links served through the Discovery Relay as relay-connected links. A relay-connected link can be thought of as similar to a link that a Discovery Proxy connects to using a USB Ethernet interface, just with a very long USB cable (that runs over TCP).

When a Discovery Proxy receives a DNS query from a DNS client via unicast, it will generate corresponding mDNS query messages on the relevant multicast link(s) for which it is acting as a proxy. For locally-connected link(s), those query messages will be sent directly. For relay-connected link(s), the query messages will be sent through the Discovery Relay that is being used to serve that multicast link.

Responses from devices on locally-connected links are processed normally. Responses from devices on relay-connected links are received by the Discovery Relay, encapsulated, and forwarded to the Client; the Client then processes these messages using the link-identifying information included in the encapsulation.

In principle it could be the case that some device is capable of performing service discovery using Multicast DNS, but not using traditional unicast DNS. Responding to mDNS queries received from the Discovery Relay could address this use case. However, continued reliance on multicast is counter to the goals of the current work in service discovery, and to benefit from wide-area service discovery such client devices should be updated to support service discovery using unicast queries.

## 8. DSO TLVs

This document defines a modest number of new DSO TLVs.

### 8.1. mDNS Link Data Request

The mDNS Link Data Request TLV conveys a link identifier from which a Client is requesting that a Discovery Relay forward mDNS traffic. The link identifier comes from the provisioning configuration (see Section 9). The DSO-TYPE for this TLV is TBD-R. DSO-LENGTH is always 5. DSO-DATA is the 8-bit address family followed by the link identifier, a 32-bit unsigned integer in network (big endian) byte order, as described in Section 9. An address family value of 1 indicates IPv4 and 2 indicates IPv6, as recorded in the IANA Registry of Address Family Numbers [AdFam].

The mDNS Link Data Request TLV can only be used as a primary TLV, and requires an acknowledgement.

At most one mDNS Link Data Request TLV may appear in a DSO message. To request multiple link subscriptions, multiple separate DSO messages are sent, each containing a single mDNS Link Data Request TLV.

A Client MUST NOT request a link if it already has an active subscription to that link on the same DSO connection. If a Discovery Relay receives a duplicate link subscription request, it MUST immediately abort that DSO session with a TCP reset (RST).

### 8.2. mDNS Link Data Discontinue

The mDNS Link Data Discontinue TLV is used by Clients to unsubscribe to mDNS messages on the specified multicast link. DSO-TYPE is TBD-D. DSO-LENGTH is always 5. DSO-DATA is the 8-bit address family followed by the 32-bit link identifier, a 32-bit unsigned integer in network (big endian) byte order, as described in Section 9.

The mDNS Link Data Discontinue TLV can only be used as a DSO unidirectional message TLV, and is not acknowledged.

At most one mDNS Link Data Discontinue TLV may appear in a DSO message. To unsubscribe from multiple links, multiple separate DSO messages are sent, each containing a single mDNS Link Data Discontinue TLV.

### 8.3. Link Identifier

This option is used both in DSO messages from Discovery Relays to Clients that contain received mDNS messages, and from Clients to Discovery Relays that contain mDNS messages to be transmitted on the multicast link. In the former case, it indicates the multicast link on which the message was received; in the latter case, it indicates the multicast link on which the message should be transmitted. DSO-TYPE is TBD-L. DSO-LENGTH is always 5. DSO-DATA is the 8-bit address family followed by the link identifier, a 32-bit unsigned integer in network (big endian) byte order, as described in Section 9.

The Link Identifier TLV can only be used as an additional TLV. The Link Identifier TLV can only appear at most once in a Discovery Relay DSO message.

### 8.4. Encapsulated mDNS Message

The Encapsulated mDNS Message TLV is used to communicate an mDNS message that a Relay is forwarding from a multicast link to a Client, or that a Client is sending to a Relay for transmission on a multicast link. Only the application-layer payload of the mDNS message is carried in the DSO "Encapsulated mDNS Message" TLV, i.e., just the DNS message itself, beginning with the DNS Message ID, not the IP or UDP headers. The DSO-TYPE for this TLV is TBD-M. DSO-LENGTH is the length of the encapsulated mDNS message. DSO-DATA is the content of the encapsulated mDNS message.

The Encapsulated mDNS Message TLV can only be used as a DSO unidirectional message TLV, and is not acknowledged.

### 8.5. IP Source

The IP Source TLV is used to report the IP source address and port from which an mDNS message was received. This TLV is present in DSO messages from Discovery Relays to Clients that contain encapsulated mDNS messages. DSO-TYPE is TBD-S. DSO-LENGTH is either 6, for an IPv4 address, or 18, for an IPv6 address. DSO-DATA is the two-byte source port, followed by the 4- or 16-byte IP Address. Both port and address are in the canonical byte order (i.e., the same representation as used in the UDP and IP packet headers, with no byte swapping).

The IP Source TLV can only be used as an additional TLV. The IP Source TLV can only appear at most once in a Discovery Relay DSO message.

### 8.6. Link State Request

The Link State Request TLV requests that the Discovery Relay report link changes. When the relay is reporting link changes and a new link becomes available, it sends a Link Available message to the Client. When a link becomes unavailable, it sends a Link Unavailable message to the Client. If there are links available when the request is received, then for each such link the relay immediately sends a Link Available Message to the Client. DSO-TYPE is TBD-P. DSO-LENGTH is 0.

The mDNS Link State Request TLV can only be used as a primary TLV, and requires an acknowledgement. The acknowledgment does not contain a Link Available TLV: it is just a response to the Link State Request message.

### 8.7. Link State Discontinue

The Link State Discontinue TLV requests that the Discovery Relay stop reporting on the availability of links supported by the relay. This cancels the effect of a Link State Request TLV. DSO-TYPE is TBD-Q. DSO-LENGTH is 0.

The mDNS Link State Discontinue TLV can only be used as a DSO unidirectional message TLV, and is not acknowledged.

### 8.8. Link Available

The Link Available TLV is used by Discovery Relays to indicate to Clients that a new link has become available. The format is the same as the Link Identifier TLV. DSO-TYPE is TBD-V. The Link Available TLV may be accompanied by one or more Link Prefix TLVs which indicate IP prefixes the Relay knows to be present on the link.

The mDNS Link Available TLV can only be used as a DSO unidirectional message TLV, and is not acknowledged.

### 8.9. Link Unavailable

The Link Unavailable TLV is used by Discovery Relays to indicate to Clients that an existing link has become unavailable. The format is the same as the Link Identifier TLV. DSO-TYPE is TBD-U.

The mDNS Link Unavailable TLV can only be used as a DSO unidirectional message TLV, and is not acknowledged.



#### 8.10. Link Prefix

The Link Prefix TLV represents an IP address or prefix configured on a link. The length is 17 for an IPv6 address or prefix, and 5 for an IPv4 address or prefix. The TLV consists of a prefix length, between 0 and 32 for IPv4 or between 0 and 128 for IPv6, represented as a single byte. This is followed by the IP address, either four or sixteen bytes. DSO-TYPE is TBD-K.

The Link Prefix TLV can only be used as a secondary TLV.

## 9. Provisioning

In order for a Discovery Proxy to use Discovery Relays, it must be configured with sufficient information to identify multicast links on which service discovery is to be supported and, if it is not running on a host that is directly connected to those multicast links, connect to Discovery Relays supporting those multicast links.

A Discovery Relay must be configured both with a set of multicast links to which the host on which it is running is connected, on which mDNS relay service is to be provided, and also with a list of one or more Clients authorized to use it.

On a network supporting DNS Service Discovery using Discovery Relays, more than one different Discovery Relay implementation may be present. While it may be that only a single Discovery Proxy is present, that implementation will need to be able to be configured to interoperate with all of the Discovery Relays that are present. Consequently, it is necessary that a standard set of configuration parameters be defined for both Discovery Proxies and Discovery Relays.

DNS Service Discovery generally operates within a constrained set of links, not across the entire internet. This section assumes that what will be configured will be a limited set of links operated by a single entity or small set of cooperating entities, among which services present on each link should be available to users on that link and every other link. This could be, for example, a home network, a small office network, or even a network covering an entire building or small set of buildings. The set of Discovery Proxies and Discovery Relays within such a network will be referred to in this section as a 'Discovery Domain'.

Depending on the context, several different candidates for configuration of Discovery Proxies and Discovery Relays may be applicable. The simplest such mechanism is a manual configuration file, but regardless of provisioning mechanism, certain configuration information needs to be communicated to the devices, as outlined below.

In the example we provide here, we only refer to configuring of IP addresses, private keys and certificates. It is also possible to use FQDNs to identify servers; this then allows for the use of DANE ([RFC8310] Section 8.2) or PKIX authentication [RFC6125]. Which method is used is to some extent up to the implementation, but at a minimum, it should be possible to associate an IP address with a self-signed certificate, and it should be possible to validate both

self-signed and PKIX-authenticated certificates, with PKIX, DANE or a pre-configured trust anchor.

#### 9.1. Provisioned Objects

Three types of objects must be described in order for Discovery Proxies and Discovery Relays to be provisioned: Discovery Proxies, Multicast Links, and Discovery Relays. "Human-readable" below means actual words or proper names that will make sense to an untrained human being. "Machine-readable" means a name that will be used by machines to identify the entity to which the name refers. Each entity must have a machine-readable name and may have a human-readable name. No two entities can have the same human-readable name. Similarly, no two entities can have the same machine-readable name.

### 9.1.1. Multicast Link

The description of a multicast link consists of:

**link-identifier** A 32-bit identifier that uniquely identifies that link within the Discovery Domain. Each link **MUST** have exactly one such identifier. Link Identifiers do not have any special semantics, and are not intended to be human-readable.

**ldh-name** A fully-qualified domain name for the multicast link that is used to form an LDH domain name as described in section 5.3 of the Discovery Proxy specification [RFC8766]. This name is used to identify the link during provisioning, and must be present.

**hr-name** A human-readable user-friendly fully-qualified domain name for the multicast link. This name **MUST** be unique within the Discovery Domain. Each multicast link **MUST** have exactly one such name. The hr-name **MAY** be the same as the ldh-name. (The hr-name is allowed to contain spaces, punctuation and rich text, but it is not required to do so.)

The ldh-name and hr-name can be used to form the LDH and human-readable domain names as described in [RFC8766], section 5.3.

Note that the ldh-name and hr-name can be used in two different ways.

On a small home network with little or no human administrative configuration, link names may be directly visible to the user. For example, a search in 'home.arpa' on a small home network may discover services on both ethernet.home.arpa and wi-fi.home.arpa. In the case of a home user who has one Ethernet-connected printer and one Wi-Fi-connected printer, discovering that they have one printer on ethernet.home.arpa and another on wi-fi.home.arpa is understandable and meaningful.

On a large corporate network with hundreds of Wi-Fi access points, the individual link names of the hundreds of multicast links are less likely to be useful to end users. In these cases, Discovery Broker functionality [I-D.sctl-discovery-broker] may be used to translate the many link names to something more meaningful to users. For example, in a building with 50 Wi-Fi access points, each with their own link names, services on all the different physical links may be presented to the user as appearing in 'headquarters.example.com'. In this case, the individual link names can be thought of similar to MAC addresses or IPv6 addresses. They are used internally by the software as unique identifiers, but generally are not exposed to end users.

### 9.1.2. Discovery Proxy

The description of a Discovery Proxy consists of:

`name` a machine-readable name used to reference this Discovery Proxy in provisioning.

`hr-name` an optional human-readable name which can appear in provisioning, monitoring and debugging systems. Must be unique within a Discovery Domain.

`certificate` a certificate that identifies the Discovery Proxy. This certificate can be shared across services on the Discovery Proxy Host. The public key in the certificate is used both to uniquely identify the Discovery Proxy and to authenticate connections from it. The certificate should be signed by its own private key.

`private-key` the private key corresponding to the public key in the certificate.

`source-ip-addresses` a list of IP addresses that may be used by the Discovery Proxy when connecting to Discovery Relays. These addresses should be addresses that are configured on the Discovery Proxy Host. They should not be temporary addresses. All such addresses must be reachable within the Discovery Domain.

`public-ip-addresses` a list of IP addresses that a Discovery Proxy listens on to receive requests from clients. This is not used for interoperation with Discovery Relays, but is mentioned here for completeness: the list of addresses listened on for incoming client requests may differ from the 'source-ip-addresses' list of addresses used for issuing outbound connection requests to Discovery Relays. If any of these addresses are reachable from outside of the Discovery Domain, services in that domain will be discoverable outside of the domain.

`multicast links` a list of multicast links on which this Discovery Proxy is expected to provide service

The private key should never be distributed to other hosts; all of the other information describing a Discovery Proxy can be safely shared with Discovery Relays.

In some configurations it may make sense for the Discovery Relay not to have a list of links, but simply to support the set of all links available on relays to which the Discovery Proxy is configured to communicate.

### 9.1.3. Discovery Relay

The description of a Discovery Relay consists of:

`name` a required machine-readable identifier used to reference the relay

`hr-name` an optional human-readable name which can appear in provisioning, monitoring and debugging systems. Must be unique within a Discovery Domain.

`certificate` a certificate that identifies the Discovery Relay. This certificate can be shared across services on the Discovery Relay Host. Indeed, if a Discovery Proxy and Discovery Relay are running on the same host, the same certificate can be used for both. The public key in the certificate uniquely identifies the Discovery Relay and is used by a Discovery Relay Client (e.g., a Discovery Proxy) to verify that it is talking to the intended Discovery Relay after a TLS connection has been established. The certificate must either be signed by its own key, or have a signature chain that can be validated using PKIX authentication [RFC6125].

`private-key` the private key corresponding to the public key in the certificate.

`listen-tuple` a list of IP address/port tuples that may be used to connect to the Discovery Relay. The relay may be configured to listen on all addresses on a single port, but this is not required, so the port as well as the address must be specified.

`multicast links` a list of multicast links to which this relay is physically connected.

The private key should never be distributed to other hosts; all of the other information describing a Discovery Relay can be safely shared with Discovery Proxies.

In some cases a Relay may not be configured with a static list of links, but may simply discover links by monitoring the set of available interfaces on the host on which the Relay is running. In that case, the relay could be configured to identify links based on the names of network interfaces, or based on the set of available prefixes seen on those interfaces. The details of this sort of configuration are not specified in this document.

## 9.2. Configuration Files

For this discussion, we assume the simplest possible means of configuring Discovery Proxies and Discovery Relays: the configuration file. Any environment where changes will happen on a regular basis will either require some automatic means of generating these configuration files as the network topology changes, or will need to use a more automatic method for configuration, such as HNCP [RFC7788].

There are many different ways to organize configuration files. This discussion assumes that multicast links, relays and proxies will be specified as objects, as described above, perhaps in a master file, and then the specific configuration of each proxy or relay will reference the set of objects in the master file, referencing objects by name. This approach is not required, but is simply shown as an example. In addition, the private keys for each proxy or relay must appear only in that proxy or relay's configuration file.

The master file contains a list of Discovery Relays, Discovery Proxies and Multicast Links. Each object has a name and all the other data associated with it. We do not formally specify the format of the file, but it might look something like this:

```
Relay upstairs
  certificate xxx
  listen-tuple 192.0.2.1 1917
  listen-tuple fd00::1 1917
  link upstairs-wifi
  link upstairs-wired
  client-allow-list main

Relay downstairs
  certificate yyy
  listen-tuple 192.51.100.1 2088
  listen-tuple fd00::2 2088
  link downstairs-wifi
  link downstairs-wired
  client-allow-list main

Proxy main
  certificate zzz
  address 203.1.113.1

Link upstairs-wifi
  id 1
  hr-name Upstairs Wifi

Link upstairs-wired
  id 2
  hr-name Upstairs Wired

Link downstairs-wifi
  id 3
  hr-name Downstairs Wifi

Link downstairs-wired
  id 4
  hr-name Downstairs Wired
```



### 9.3. Discovery Proxy Private Configuration

The Discovery Proxy configuration contains enough information to identify which Discovery Proxy is being configured, enumerate the list of multicast links it is intended to serve, and provide keying information it can use to authenticate to Discovery Relays. It may also contain custom information about the port and/or IP address(es) on which it will respond to DNS queries.

An example configuration, following the convention used in this section, might look something like this:

```
Proxy main
  private-key zzz
  subscribe upstairs-wifi
  subscribe downstairs-wifi
  subscribe upstairs-wired
  subscribe downstairs-wired
```

When combined with the master file, this configuration is sufficient for the Discovery Proxy to identify and connect to the Discovery Relays that serve the links it is configured to support.

### 9.4. Discovery Relay Private Configuration

The Discovery Relay configuration just needs to tell the Discovery Relay what name to use to find its configuration in the master file, and what the private key is corresponding to its certificate (public key) in the master file. For example:

```
Relay Downstairs
  private-key yyy
```

## 10. Security Considerations

Part of the purpose of the Multicast DNS Discovery Relay protocol is to place a simple relay, analogous to a BOOTP relay, into routers and similar devices that may not be updated frequently. The BOOTP [RFC0951] protocol has been around since 1985, and continues to be useful today. The BOOTP protocol uses no encryption, and in many enterprise networks this is considered acceptable. In contrast, the Discovery Relay protocol requires TLS 1.3. A concern is that after 20 or 30 years, TLS 1.3, or some of the encryption algorithms it uses, may become obsolete, rendering devices that require it unusable. Our assessment is that TLS 1.3 probably will be around for many years to come. TLS 1.0 [RFC2246] was used for about a decade, and similarly TLS 1.2 [RFC5246] was also used for about a decade. We expect TLS 1.3 [RFC8446] to have at least that lifespan. In addition, recent IETF efforts are pushing for better software update practices for devices like routers, for other security reasons, making it likely that in ten years time it will be less common to be using routers that haven't had a software update for ten years. However, authors of encryption specifications and libraries should be aware of the potential backwards compatibility issues if an encryption algorithm becomes deprecated. This specification RECOMMENDS that if an encryption algorithm becomes deprecated, then rather than remove that encryption algorithm entirely, encryption libraries should disable that encryption algorithm by default, but leave the code present with an option for client software to enable it in special cases, such as a recent Client talking to an ancient Discovery Relay. Using no encryption, like BOOTP, would eliminate this backwards compatibility concern, but we feel that in such a future hypothetical scenario, using even a weak encryption algorithm still makes passive eavesdropping and tampering harder, and is preferable to using no encryption at all.

## 11. IANA Considerations

The IANA is kindly requested to update the DSO Type Codes Registry [RFC8490] by allocating codes for each of the TBD type codes listed in the following table, and by updating this document, here and in Section 8. Each type code should list this document as its reference document.

DSO-TYPE	Status	Name
TBD-R	Standard	Link Data Request
TBD-D	Standard	Link Data Discontinue
TBD-L	Standard	Link Identifier
TBD-M	Standard	Encapsulated mDNS Message
TBD-S	Standard	IP Source
TBD-P	Standard	Link State Request
TBD-Q	Standard	Link State Discontinue
TBD-V	Standard	Link Available
TBD-U	Standard	Link Unavailable
TBD-K	Standard	Link Prefix

DSO Type Codes to be allocated

## 12. Acknowledgments

Thanks to Derek Atkins for the secdir early review.

## 13. References

### 13.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, <<https://www.rfc-editor.org/info/rfc7323>>.
- [RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016, <<https://www.rfc-editor.org/info/rfc7766>>.
- [RFC7788] Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", RFC 7788, DOI 10.17487/RFC7788, April 2016, <<https://www.rfc-editor.org/info/rfc7788>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8310] Dickinson, S., Gillmor, D., and T. Reddy, "Usage Profiles for DNS over TLS and DNS over DTLS", RFC 8310, DOI 10.17487/RFC8310, March 2018, <<https://www.rfc-editor.org/info/rfc8310>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8490] Bellis, R., Cheshire, S., Dickinson, J., Dickinson, S., Lemon, T., and T. Pusateri, "DNS Stateful Operations", RFC 8490, DOI 10.17487/RFC8490, March 2019, <<https://www.rfc-editor.org/info/rfc8490>>.
- [RFC8766] Cheshire, S., "Discovery Proxy for Multicast DNS-Based Service Discovery", RFC 8766, DOI 10.17487/RFC8766, June 2020, <<https://www.rfc-editor.org/info/rfc8766>>.

### 13.2. Informative References

- [AdFam] "IANA Address Family Numbers Registry", <<https://www.iana.org/assignments/address-family-numbers/>>.
- [AdProx] Cheshire, S. and T. Lemon, "Advertising Proxy for DNS-SD Service Registration Protocol", draft-sctl-advertising-proxy-00 (work in progress), July 2020.
- [I-D.ietf-mboned-ieee802-mcast-problems] Perkins, C., McBride, M., Stanley, D., Kumari, W., and J. Zuniga, "Multicast Considerations over IEEE 802 Wireless Media", draft-ietf-mboned-ieee802-mcast-problems-12 (work in progress), October 2020.
- [I-D.sctl-discovery-broker] Cheshire, S. and T. Lemon, "Service Discovery Broker", draft-sctl-discovery-broker-00 (work in progress), July 2017.
- [NOTSENT] Dumazet, E., "TCP\_NOTSENT\_LOWAT socket option", July 2013, <<https://lwn.net/Articles/560082/>>.

- [PRIO] Chan, W., "Prioritization Only Works When There's Pending Data to Prioritize", January 2014, <<https://insouciant.org/tech/prioritization-only-works-when-theres-pending-data-to-prioritize/>>.
- [RFC0951] Croft, W. and J. Gilmore, "Bootstrap Protocol", RFC 951, DOI 10.17487/RFC0951, September 1985, <<https://www.rfc-editor.org/info/rfc951>>.
- [RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, DOI 10.17487/RFC2246, January 1999, <<https://www.rfc-editor.org/info/rfc2246>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [TR-069] Broadband Forum, "CPE WAN Management Protocol", November 2013, <[https://www.broadband-forum.org/technical/download/TR-069\\_Amendment-5.pdf](https://www.broadband-forum.org/technical/download/TR-069_Amendment-5.pdf)>.

#### Authors' Addresses

Ted Lemon  
Apple Inc.  
One Apple Park Way  
Cupertino, California 95014  
United States of America

Phone: +1 (408) 996-1010  
Email: [elemen@apple.com](mailto:elemen@apple.com)

Stuart Cheshire  
Apple Inc.  
One Apple Park Way  
Cupertino, California 95014  
United States of America

Phone: +1 (408) 996-1010  
Email: [cheshire@apple.com](mailto:cheshire@apple.com)

Internet Engineering Task Force  
Internet-Draft  
Intended status: Standards Track  
Expires: 26 October 2022

T. Lemon  
S. Cheshire  
Apple Inc.  
24 April 2022

Service Registration Protocol for DNS-Based Service Discovery  
draft-ietf-dnssd-srp-13

## Abstract

The Service Registration Protocol for DNS-Based Service Discovery uses the standard DNS Update mechanism to enable DNS-Based Service Discovery using only unicast packets. This makes it possible to deploy DNS Service Discovery without multicast, which greatly improves scalability and improves performance on networks where multicast service is not an optimal choice, particularly 802.11 (Wi-Fi) and 802.15.4 (IoT) networks. DNS-SD Service registration uses public keys and SIG(0) to allow services to defend their registrations against attack.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 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 material or to cite them other than as "work in progress."

This Internet-Draft will expire on 26 October 2022.

## Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Service Registration Protocol . . . . .	5
2.1. Protocol Variants . . . . .	5
2.1.1. Full-featured Hosts . . . . .	5
2.1.2. Constrained Hosts . . . . .	6
2.1.3. Why two variants? . . . . .	6
2.2. Protocol Details . . . . .	7
2.2.1. What to publish . . . . .	7
2.2.2. Where to publish it . . . . .	7
2.2.3. How to publish it . . . . .	8
2.2.3.1. How DNS-SD Service Registration differs from standard RFC2136 DNS Update . . . . .	8
2.2.4. How to secure it . . . . .	9
2.2.4.1. First-Come First-Served Naming . . . . .	9
2.2.5. Service Behavior . . . . .	9
2.2.5.1. Public/Private key pair generation and storage . . . . .	9
2.2.5.2. Name Conflict Handling . . . . .	10
2.2.5.3. Record Lifetimes . . . . .	10
2.2.5.4. Compression in SRV records . . . . .	11
2.2.5.5. Removing published services . . . . .	11
2.3. Validation and Processing of SRP Updates . . . . .	12
2.3.1. Validation of Adds and Deletes . . . . .	12
2.3.1.1. Service Discovery Instruction . . . . .	13
2.3.1.2. Service Description Instruction . . . . .	13
2.3.1.3. Host Description Instruction . . . . .	14
2.3.2. Valid SRP Update Requirements . . . . .	14
2.3.3. FCFS Name And Signature Validation . . . . .	15
2.3.4. Handling of Service Subtypes . . . . .	16
2.3.5. SRP Update response . . . . .	16
2.3.6. Optional Behavior . . . . .	16
3. TTL Consistency . . . . .	17
4. Maintenance . . . . .	18
4.1. Cleaning up stale data . . . . .	18
5. Security Considerations . . . . .	19
5.1. Source Validation . . . . .	19
5.2. SRP Server Authentication . . . . .	20
5.3. Required Signature Algorithm . . . . .	20
6. Privacy Considerations . . . . .	21
7. Delegation of 'service.arpa.' . . . . .	21
8. IANA Considerations . . . . .	21
8.1. Registration and Delegation of 'service.arpa' as a Special-Use Domain Name . . . . .	21



8.2. 'dnssd-srp' Service Name . . . . .	21
8.3. 'dnssd-srp-tls' Service Name . . . . .	22
8.4. Anycast Address . . . . .	22
9. Implementation Status . . . . .	23
10. Acknowledgments . . . . .	24
11. Normative References . . . . .	24
12. Informative References . . . . .	26
Appendix A. Testing using standard RFC2136-compliant servers . .	27
Appendix B. How to allow services to update standard RFC2136-compliant servers . . . . .	28
Appendix C. Sample BIND9 configuration for default.service.arpa. . . . .	28
Authors' Addresses . . . . .	29

## 1. Introduction

DNS-Based Service Discovery [RFC6763] is a component of Zero Configuration Networking [RFC6760] [ZC] [I-D.cheshire-dnssd-roadmap].

This document describes an enhancement to DNS-Based Service Discovery [RFC6763] that allows services to register their services using the DNS protocol rather than using Multicast DNS [RFC6762] (mDNS). There is already a large installed base of DNS-SD clients that can discover services using the DNS protocol.

This document is intended for three audiences: implementors of software that provides services that should be advertised using DNS-SD, implementors of DNS servers that will be used in contexts where DNS-SD registration is needed, and administrators of networks where DNS-SD service is required. The document is intended to provide sufficient information to allow interoperable implementation of the registration protocol.

DNS-Based Service Discovery (DNS-SD) allows services to advertise the fact that they provide service, and to provide the information required to access that service. DNS-SD clients can then discover the set of services of a particular type that are available. They can then select a service from among those that are available and obtain the information required to use it. Although DNS-SD using the DNS protocol (as opposed to mDNS) can be more efficient and versatile, it is not common in practice, because of the difficulties associated with updating authoritative DNS services with service information.

Existing practice for updating DNS zones is to either manually enter new data, or else use DNS Update [RFC2136]. Unfortunately DNS Update requires either that the authoritative DNS server automatically trust updates, or else that the DNS Update client have some kind of shared

secret or public key that is known to the DNS server and can be used to authenticate the update. Furthermore, DNS Update can be a fairly chatty process, requiring multiple round trips with different conditional predicates to complete the update process.

The SRP protocol adds a set of default heuristics for processing DNS updates that eliminates the need for DNS update conditional predicates: instead, the SRP server has a set of default predicates that are applied to the update, and the update either succeeds entirely, or fails in a way that allows the registering service to know what went wrong and construct a new update.

SRP also adds a feature called First-Come, First-Served Naming, which allows the registering service to claim a name that is not yet in use, and, using SIG(0) [RFC2931], to authenticate both the initial claim and subsequent updates. This prevents name conflicts, since a second SRP service attempting to claim the same name will not possess the SIG(0) key used by the first service to claim it, and so its claim will be rejected and the second service will have to choose a new name.

Finally, SRP adds the concept of a 'lease,' similar to leases in Dynamic Host Configuration Protocol [RFC8415]. The SRP registration itself has a lease which may be on the order of an hour; if the registering service does not renew the lease before it has elapsed, the registration is removed. The claim on the name can have a longer lease, so that another service cannot claim the name, even though the registration has expired.

The Service Registration Protocol for DNS-SD (SRP), described in this document, provides a reasonably secure mechanism for publishing this information. Once published, these services can be readily discovered by DNS-SD clients using standard DNS lookups.

The DNS-SD specification [RFC6763], Section 10 ("Populating the DNS with Information"), briefly discusses ways that services can publish their information in the DNS namespace. In the case of mDNS, it allows services to publish their information on the local link, using names in the ".local" namespace, which makes their services directly discoverable by peers attached to that same local link.

RFC6763 also allows clients to discover services using the DNS protocol [RFC1035]. This can be done by having a system administrator manually configure service information in the DNS, but manually populating DNS authoritative server databases is costly and potentially error-prone, and requires a knowledgeable network administrator. Consequently, although all DNS-SD client implementations of which we are aware support DNS-SD using DNS queries, in practice it is used much less frequently than mDNS.

The Discovery Proxy [RFC8766] provides one way to automatically populate the DNS namespace, but is only appropriate on networks where services are easily advertised using mDNS. This document describes a solution more suitable for networks where multicast is inefficient, or where sleepy devices are common, by supporting both offering of services, and discovery of services, using unicast.

## 2. Service Registration Protocol

Services that implement SRP use DNS Update [RFC2136] [RFC3007] to publish service information in the DNS. Two variants exist, one for full-featured hosts, and one for devices designed for "Constrained-Node Networks" [RFC7228]. An SRP server is most likely an authoritative DNS server, or else is updating an authoritative DNS server. There is no requirement that the server that is receiving SRP requests be the same server that is answering queries that return records that have been registered.

### 2.1. Protocol Variants

#### 2.1.1. Full-featured Hosts

Full-featured hosts are either configured manually with a registration domain, or use the "dr.\_dns-sd.\_udp.<domain>" query ([RFC6763], Section 11) to learn the default registration domain from the network. RFC6763 says to discover the registration domain using either ".local" or a network-supplied domain name for <domain>. Services using SRP MUST use the domain name received through the DHCPv4 Domain Name option ([RFC2132], Section 3.17), if available, or the Neighbor Discovery DNS Search List option [RFC8106]. If the DNS Search List option contains more than one domain name, it MUST NOT be used. If neither option is available, the Service Registration protocol is not available on the local network.

Manual configuration of the registration domain can be done either by querying the list of available registration zones ("r.\_dns-sd.\_udp") and allowing the user to select one from the UI, or by any other means appropriate to the particular use case being addressed. Full-featured devices construct the names of the SRV, TXT, and PTR records

describing their service(s) as subdomains of the chosen service registration domain. For these names they then discover the zone apex of the closest enclosing DNS zone using SOA queries [RFC8765]. Having discovered the enclosing DNS zone, they query for the "\_dnssd-srp.\_tcp.<zone>" SRV record to discover the server to which they should send DNS updates. Hosts that support SRP Updates using TLS use the "\_dnssd-srp-tls.\_tcp.<zone>" SRV record instead.

#### 2.1.2. Constrained Hosts

For devices designed for Constrained-Node Networks [RFC7228] some simplifications are available. Instead of being configured with (or discovering) the service registration domain, the (proposed) special-use domain name (see [RFC6761]) "default.service.arpa" is used. The details of how SRP server(s) are discovered will be specific to the constrained network, and therefore we do not suggest a specific mechanism here.

SRP clients on constrained networks are expected to receive from the network a list of SRP servers with which to register. It is the responsibility of a Constrained-Node Network supporting SRP to provide one or more SRP server addresses. It is the responsibility of the SRP server supporting a Constrained-Node Network to handle the updates appropriately. In some network environments, updates may be accepted directly into a local "default.service.arpa" zone, which has only local visibility. In other network environments, updates for names ending in "default.service.arpa" may be rewritten internally to names with broader visibility.

#### 2.1.3. Why two variants?

The reason for these different assumptions is that low-power devices that typically use Constrained-Node Networks may have very limited battery power. The series of DNS lookups required to discover an SRP server and then communicate with it will increase the power required to advertise a service; for low-power devices, the additional flexibility this provides does not justify the additional use of power. It is also fairly typical of such networks that some network service information is obtained as part of the process of joining the network, and so this can be relied upon to provide nodes with the information they need.

Networks that are not constrained networks can have more complicated topologies at the Internet layer. Nodes connected to such networks can be assumed to be able to do DNSSD service registration domain discovery. Such networks are generally able to provide registration domain discovery and routing. By requiring the use of TCP, the possibility of off-network spoofing is eliminated.

## 2.2. Protocol Details

We will discuss several parts to this process: how to know what to publish, how to know where to publish it (under what name), how to publish it, how to secure its publication, and how to maintain the information once published.

### 2.2.1. What to publish

We refer to the DNS Update message sent by services using SRP as an SRP Update. Three types of updates appear in an SRP update: Service Discovery records, Service Description records, and Host Description records.

- \* Service Discovery records are one or more PTR RRs, mapping from the generic service type (or subtype) to the specific Service Instance Name.
- \* Service Description records are exactly one SRV RR, exactly one KEY RR, and one or more TXT RRs, all with the same name, the Service Instance Name ([RFC6763], Section 4.1). In principle Service Description records can include other record types, with the same Service Instance Name, though in practice they rarely do. The Service Instance Name MUST be referenced by one or more Service Discovery PTR records, unless it is a placeholder service registration for an intentionally non-discoverable service name.
- \* The Host Description records for a service are a KEY RR, used to claim exclusive ownership of the service registration, and one or more RRs of type A or AAAA, giving the IPv4 or IPv6 address(es) of the host where the service resides.

[RFC6763] describes the details of what each of these types of updates contains, with the exception of the KEY RR, which is defined in [RFC2539]. These RFCs should be considered the definitive source for information about what to publish; the reason for summarizing this here is to provide the reader with enough information about what will be published that the service registration process can be understood at a high level without first learning the full details of DNS-SD. Also, the "Service Instance Name" is an important aspect of first-come, first-serve naming, which we describe later on in this document.

### 2.2.2. Where to publish it

Multicast DNS uses a single namespace, ".local", which is valid on the local link. This convenience is not available for DNS-SD using the DNS protocol: services must exist in some specific unicast namespace.

As described above, full-featured devices are responsible for knowing in what domain they should register their services. Devices made for Constrained-Node Networks register in the (proposed) special use domain name [RFC6761] "default.service.arpa", and let the SRP server handle rewriting that to a different domain if necessary.

### 2.2.3. How to publish it

It is possible to issue a DNS Update that does several things at once; this means that it's possible to do all the work of adding a PTR resource record to the PTR RRset on the Service Name, and creating or updating the Service Instance Name and Host Description, in a single transaction.

An SRP Update takes advantage of this: it is implemented as a single DNS Update message that contains a service's Service Discovery records, Service Description records, and Host Description records.

Updates done according to this specification are somewhat different than regular DNS Updates as defined in RFC2136. The RFC2136 update process can involve many update attempts: you might first attempt to add a name if it doesn't exist; if that fails, then in a second message you might update the name if it does exist but matches certain preconditions. Because the registration protocol uses a single transaction, some of this adaptability is lost.

In order to allow updates to happen in a single transaction, SRP Updates do not include update prerequisites. The requirements specified in Section 2.3 are implicit in the processing of SRP Updates, and so there is no need for the service sending the SRP Update to put in any explicit prerequisites.

#### 2.2.3.1. How DNS-SD Service Registration differs from standard RFC2136 DNS Update

DNS-SD Service Registration is based on standard RFC2136 DNS Update, with some differences:

- \* It implements first-come first-served name allocation, protected using SIG(0) [RFC2931].
- \* It enforces policy about what updates are allowed.
- \* It optionally performs rewriting of "default.service.arpa" to some other domain.
- \* It optionally performs automatic population of the address-to-name reverse mapping domains.
- \* An SRP server is not required to implement general DNS Update prerequisite processing.

- \* Constrained-Node SRP clients are allowed to send updates to the generic domain "default.service.arpa"

#### 2.2.4. How to secure it

Traditional DNS update is secured using the TSIG protocol, which uses a secret key shared between the DNS Update client (which issues the update) and the server (which authenticates it). This model does not work for automatic service registration.

The goal of securing the DNS-SD Registration Protocol is to provide the best possible security given the constraint that service registration has to be automatic. It is possible to layer more operational security on top of what we describe here, but what we describe here is an improvement over the security of mDNS. The goal is not to provide the level of security of a network managed by a skilled operator.

##### 2.2.4.1. First-Come First-Served Naming

First-Come First-Serve naming provides a limited degree of security: a service that registers its service using DNS-SD Registration protocol is given ownership of a name for an extended period of time based on the key used to authenticate the DNS Update. As long as the registration service remembers the name and the key used to register that name, no other service can add or update the information associated with that. FCFS naming is used to protect both the Service Description and the Host Description.

#### 2.2.5. Service Behavior

##### 2.2.5.1. Public/Private key pair generation and storage

The service generates a public/private key pair. This key pair **MUST** be stored in stable storage; if there is no writable stable storage on the SRP client, the SRP client **MUST** be pre-configured with a public/private key pair in read-only storage that can be used. This key pair **MUST** be unique to the device. A device with rewritable storage should retain this key indefinitely. When the device changes ownership, it may be appropriate to erase the old key and install a new one. Therefore, the SRP client on the device **SHOULD** provide a mechanism to overwrite the key, for example as the result of a "factory reset."

When sending DNS updates, the service includes a KEY record containing the public portion of the key in each Host Description Instruction and each Service Description Instruction. Each KEY record **MUST** contain the same public key. The update is signed using

SIG(0), using the private key that corresponds to the public key in the KEY record. The lifetimes of the records in the update is set using the EDNS(0) Update Lease option [I-D.sekar-dns-ul].

The KEY record in Service Description updates MAY be omitted for brevity; if it is omitted, the SRP server MUST behave as if the same KEY record that is given for the Host Description is also given for each Service Description for which no KEY record is provided. Omitted KEY records are not used when computing the SIG(0) signature.

#### 2.2.5.2. Name Conflict Handling

Both Host Description records and Service Description Records can have names that result in name conflicts. Service Discovery records cannot have name conflicts. If any Host Description or Service Description record is found by the server to have a conflict with an existing name, the server will respond to the SRP Update with a YXDOMAIN rcode. In this case, the Service MUST either abandon the service registration attempt, or else choose a new name.

There is no specific requirement for how this is done; typically, however, the service will append a number to the preferred name. This number could be sequentially increasing, or could be chosen randomly. One existing implementation attempts several sequential numbers before choosing randomly. So for instance, it might try host.service.arpa, then host-1.service.arpa, then host-2.service.arpa, then host-31773.service.arpa.

#### 2.2.5.3. Record Lifetimes

The lifetime of the DNS-SD PTR, SRV, A, AAAA and TXT records [RFC6763] uses the LEASE field of the Update Lease option, and is typically set to two hours. This means that if a device is disconnected from the network, it does not appear in the user interfaces of devices looking for services of that type for too long.

The lifetime of the KEY records is set using the KEY-LEASE field of the Update Lease Option, and should be set to a much longer time, typically 14 days. The result of this is that even though a device may be temporarily unplugged, disappearing from the network for a few days, it makes a claim on its name that lasts much longer.

This means that even if a device is unplugged from the network for a few days, and its services are not available for that time, no other device can come along and claim its name the moment it disappears from the network. In the event that a device is unplugged from the network and permanently discarded, then its name is eventually cleaned up and made available for re-use.



#### 2.2.5.4. Compression in SRV records

Although [RFC2782] requires that the target name in the SRV record not be compressed, an SRP client SHOULD compress the target in the SRV record. The motivation for not compressing in RFC2782 is not stated, but is assumed to be because a caching resolver that does not understand the format of the SRV record might store it as binary data and thus return an invalid pointer in response to a query. This does not apply in the case of SRP: an SRP server needs to understand SRV records in order to validate the SRP Update. Compression of the target potentially saves substantial space in the SRP Update.

#### 2.2.5.5. Removing published services

##### 2.2.5.5.1. Removing all published services

To remove all the services registered to a particular host, the SRP client retransmits its most recent update with an Update Lease option that has a LEASE value of zero. If the registration is to be permanently removed, KEY-LEASE should also be zero. Otherwise, it should have the same value it had previously; this holds the name in reserve for when the SRP client is once again able to provide the service.

SRP clients are normally expected to remove all service instances when removing a host. However, in some cases a SRP client may not have retained sufficient state to know that some service instance is pointing to a host that it is removing. This method of removing services is intended for the case where the client is going offline and does not want its services advertised. Therefore, it is sufficient for the client to send the Host Description Instruction (Section 2.3.1.3).

To support this, when removing services based on the lease time being zero, an SRP server MUST remove all service instances pointing to a host when a host is removed, even if the SRP client doesn't list them explicitly. If the key lease time is nonzero, the SRP server MUST NOT delete the KEY records for these SRP clients.

##### 2.2.5.5.2. Removing some published services

In some use cases a client may need to remove some specific service, without removing its other services. This can be accomplished in one of two ways. To simply remove a specific service, the client sends a valid SRP Update where the Service Discovery Instruction (Section 2.3.1.1) contains a single Delete an RR from an RRset ([RFC2136], Section 2.5.4) update that deletes the PTR record whose target is the service instance name. The Service Description

Instruction (Section 2.3.1.2) in this case contains a single Delete all RRsets from a Name ([RFC2136], Section 2.5.3) update to the service instance name.

The second alternative is used when some service is being replaced by a different service with a different service instance name. In this case, the old service is deleted as in the first alternative. The new service is added, just as it would be in an update that wasn't deleting the old service. Because both the removal of the old service and the add of the new service consist of a valid Service Discovery Instruction and a valid Service Description Instruction, the update as a whole is a valid SRP Update, and will result in the old service being removed and the new one added, or, to put it differently, in the old service being replaced by the new service.

It is perhaps worth noting that if a service is being updated without the service instance name changing, that will look very much like the second alternative above. The difference is that because the target for the PTR record in the Service Discovery Instruction is the same for both the Delete An RR From An RRset update and the Add To An RRset update, these will be seen as a single Service Description Instruction, not as two Instructions. The same would be true of the Service Description Instruction.

Whichever of these two alternatives is used, the host lease will be updated with the lease time provided in the SRP update. In neither of these cases is it permissible to delete the host. All services must point to a host. If a host is to be deleted, this must be done using the method described in Section 2.2.5.5.1, which deletes the host and all services that have that host as their target.

## 2.3. Validation and Processing of SRP Updates

### 2.3.1. Validation of Adds and Deletes

The SRP server first validates that the DNS Update is a syntactically and semantically valid DNS Update according to the rules specified in RFC2136.

SRP Updates consist of a set of `_instructions_` that together add or remove one or more services. Each instruction consists of some combination of delete updates and add updates. When an instruction contains a delete and an add, the delete **MUST** precede the add.

The SRP server checks each instruction in the SRP Update to see that it is either a Service Discovery Instruction, a Service Description Instruction, or a Host Description Instruction. Order matters in DNS updates. Specifically, deletes must precede adds for records that

the deletes would affect; otherwise the add will have no effect. This is the only ordering constraint; aside from this constraint, updates may appear in whatever order is convenient when constructing the update.

Because the SRP Update is a DNS update, it MUST contain a single question that indicates the zone to be updated. Every delete and update in an SRP Update MUST be within the zone that is specified for the SRP Update.

#### 2.3.1.1. Service Discovery Instruction

An instruction is a Service Discovery Instruction if it contains

- \* exactly one "Add to an RRSet" or exactly one "Delete an RR from an RRSet" ([RFC2136], Section 2.5.1) RR update,
- \* which updates a PTR RR,
- \* the target of which is a Service Instance Name
- \* for which name a Service Description Instruction is present in the SRP Update
- \* if the Service Discovery Instruction is an "Add to an RRSet" instruction, the Service Description Instruction does not match if it does not contain an "Add to an RRset" update for the SRV RR describing that service.
- \* if the Service Discovery Instruction is a "Delete an RR from an RRSet" update, the Service Description Instruction does not match if it contains an "Add to an RRset" update.
- \* Service Discovery Instructions do not contain any other add or delete updates.

#### 2.3.1.2. Service Description Instruction

An instruction is a Service Description Instruction if, for the appropriate Service Instance Name, it contains

- \* exactly one "Delete all RRsets from a name" update for the service instance name ([RFC2136], Section 2.5.3),
- \* zero or one "Add to an RRset" SRV RR,
- \* zero or one "Add to an RRset" KEY RR that, if present, contains the public key corresponding to the private key that was used to sign the message (if present, the KEY MUST match the KEY RR given in the Host Description),
- \* zero or more "Add to an RRset" TXT RRs,
- \* If there is one "Add to an RRset" SRV update, there MUST be at least one "Add to an RRset" TXT update.
- \* the target of the SRV RR Add, if present points to a hostname for which there is a Host Description Instruction in the SRP Update, or

- \* if there is no "Add to an RRset" SRV RR, then either
  - the name to which the "Delete all RRsets from a name" applies does not exist, or
  - there is an existing KEY RR on that name, which matches the key with which the SRP Update was signed.
- \* Service Descriptions Instructions do not modify any other resource records.

An SRP server MUST correctly handle compressed names in the SRV target.

#### 2.3.1.3. Host Description Instruction

An instruction is a Host Description Instruction if, for the appropriate hostname, it contains

- \* exactly one "Delete all RRsets from a name" RR,
- \* one or more "Add to an RRset" RRs of type A and/or AAAA,
- \* A and/or AAAA records must be of sufficient scope to be reachable by all hosts that might query the DNS. If a link-scope address or IPv4 autoconfiguration address is provided by the SRP client, the SRP server MUST treat this as if no address records were received; that is, the Host Description is not valid.
- \* exactly one "Add to an RRset" RR that adds a KEY RR that contains the public key corresponding to the private key that was used to sign the message,
- \* there is a Service Instance Name Instruction in the SRP Update for which the SRV RR that is added points to the hostname being updated by this update.
- \* Host Description Instructions do not modify any other resource records.

#### 2.3.2. Valid SRP Update Requirements

An SRP Update MUST include zero or more Service Discovery Instructions. For each Service Discovery Instruction, there MUST be at least one Service Description Instruction. Note that in the case of SRP subtypes (Section 7.1 of [RFC6763]), it's quite possible that two Service Discovery Instructions might reference the same Service Description Instruction. For each Service Description Instruction there MUST be at least one Service Discovery Instruction with its service instance name as the target of its PTR record. There MUST be exactly one Host Description Instruction. Every Service Description Instruction must have that Host Description Instruction as the target of its SRV record. A DNS Update that does not meet these constraints is not an SRP Update.

A DNS Update that contains any additional adds or deletes that cannot be identified as Service Discovery, Service Description or Host Description Instructions is not an SRP Update. A DNS update that contains any prerequisites is not an SRP Update. Such messages should either be processed as regular RFC2136 updates, including access control checks and constraint checks, if supported, or else rejected with RCODE=REFUSED.

In addition, in order for an update to be a valid SRP Update, the target of every Service Discovery Instruction MUST be a Service Description Instruction that is present in the SRP Update. There MUST NOT be any Service Description Instruction to which no Service Discovery Instruction points. The target of the SRV record in every Service Description Instruction MUST be the single Host Description Instruction.

If the definitions of each of these instructions are followed carefully and the update requirements are validated correctly, many DNS Updates that look very much like SRP Updates nevertheless will fail to validate. For example, a DNS update that contains an Add to an RRset instruction for a Service Name and an Add to an RRset instruction for a Service Instance Name, where the PTR record added to the Service Name does not reference the Service Instance Name, is not a valid SRP Update message, but may be a valid RFC2136 update.

### 2.3.3. FCFS Name And Signature Validation

Assuming that a DNS Update message has been validated with these conditions and is a valid SRP Update, the server checks that the name in the Host Description Instruction exists. If so, then the server checks to see if the KEY record on that name is the same as the KEY record in the Host Description Instruction. The server performs the same check for the KEY records in any Service Description Instructions. For KEY records that were omitted from Service Description Instructions, the KEY from the Host Description Instruction is used. If any existing KEY record corresponding to a KEY record in the SRP Update does not match the KEY record in the SRP Update (whether provided or taken from the Host Description Instruction), then the server MUST reject the SRP Update with the YXDOMAIN RCODE.

Otherwise, the server validates the SRP Update using SIG(0) against the public key in the KEY record of the Host Description Instruction. If the validation fails, the server MUST reject the SRP Update with the REFUSED RCODE. Otherwise, the SRP Update is considered valid and authentic, and is processed according to the method described in RFC2136.

KEY record updates omitted from Service Description Instruction are processed as if they had been explicitly present: every Service Description that is updated MUST, after the SRP Update has been applied, have a KEY RR, and it must be the same KEY RR that is present in the Host Description to which the Service Description refers.

#### 2.3.4. Handling of Service Subtypes

SRP servers MUST treat the update instructions for a service type and all its subtypes as atomic. That is, when a service and its subtypes are being updated, whatever information appears in the SRP Update is the entirety of information about that service and its subtypes. If any subtype appeared in a previous update but does not appear in the current update, then the DNS server MUST remove that subtype.

Similarly, there is no mechanism for deleting subtypes. A delete of a service deletes all of its subtypes. To delete an individual subtype, an SRP Update must be constructed that contains the service type and all subtypes for that service.

#### 2.3.5. SRP Update response

The status that is returned depends on the result of processing the update, and can be either SUCCESS or SERVFAIL: all other possible outcomes should already have been accounted for when applying the constraints that qualify the update as an SRP Update.

#### 2.3.6. Optional Behavior

The server MAY add a Reverse Mapping that corresponds to the Host Description. This is not required because the Reverse Mapping serves no protocol function, but it may be useful for debugging, e.g. in annotating network packet traces or logs. In order for the server to add a reverse mapping update, it must be authoritative for the zone or have credentials to do the update. The SRP client MAY also do a reverse mapping update if it has credentials to do so.

The server MAY apply additional criteria when accepting updates. In some networks, it may be possible to do out-of-band registration of keys, and only accept updates from pre-registered keys. In this case, an update for a key that has not been registered should be rejected with the REFUSED RCODE.

There are at least two benefits to doing this rather than simply using normal SIG(0) DNS updates. First, the same registration protocol can be used in both cases, so both use cases can be addressed by the same service implementation. Second, the registration protocol includes maintenance functionality not present with normal DNS updates.

Note that the semantics of using SRP in this way are different than for typical RFC2136 implementations: the KEY used to sign the SRP Update only allows the SRP client to update records that refer to its Host Description. RFC2136 implementations do not normally provide a way to enforce a constraint of this type.

The server may also have a dictionary of names or name patterns that are not permitted. If such a list is used, updates for Service Instance Names that match entries in the dictionary are rejected with YXDOMAIN.

### 3. TTL Consistency

All RRs within an RRset are required to have the same TTL (Clarifications to the DNS Specification [RFC2181], Section 5.2). In order to avoid inconsistencies, SRP places restrictions on TTLs sent by services and requires that SRP servers enforce consistency.

Services sending SRP Updates MUST use consistent TTLs in all RRs within the SRP Update.

SRP servers MUST check that the TTLs for all RRs within the SRP Update are the same. If they are not, the SRP update MUST be rejected with a REFUSED RCODE.

Additionally, when adding RRs to an RRset, for example when processing Service Discovery records, the server MUST use the same TTL on all RRs in the RRset. How this consistency is enforced is up to the implementation.

TTLs sent in SRP Updates are advisory: they indicate the SRP client's guess as to what a good TTL would be. SRP servers may override these TTLs. SRP servers SHOULD ensure that TTLs are reasonable: neither too long nor too short. The TTL should never be longer than the lease time (Section 4.1). Shorter TTLs will result in more frequent data refreshes; this increases latency on the DNS-SD client side, increases load on any caching resolvers and on the authoritative server, and also increases network load, which may be an issue for constrained networks. Longer TTLs will increase the likelihood that data in caches will be stale. TTL minimums and maximums SHOULD be configurable by the operator of the SRP server.

## 4. Maintenance

### 4.1. Cleaning up stale data

Because the DNS-SD registration protocol is automatic, and not managed by humans, some additional bookkeeping is required. When an update is constructed by the SRP client, it **MUST** include an EDNS(0) Update Lease Option [I-D.sekar-dns-ul]. The Update Lease Option contains two lease times: the Lease Time and the Key Lease Time.

These leases are promises, similar to DHCP leases [RFC2131], from the SRP client that it will send a new update for the service registration before the lease time expires. The Lease time is chosen to represent the time after the update during which the registered records other than the KEY record should be assumed to be valid. The Key Lease time represents the time after the update during which the KEY record should be assumed to be valid.

The reasoning behind the different lease times is discussed in the section on first-come, first-served naming (Section 2.2.4.1). SRP servers may be configured with limits for these values. A default limit of two hours for the Lease and 14 days for the SIG(0) KEY are currently thought to be good choices. Constrained devices with limited battery that wake infrequently are likely to request longer leases; servers that support such devices may need to set higher limits. SRP clients that are going to continue to use names on which they hold leases should update well before the lease ends, in case the registration service is unavailable or under heavy load.

The lease time applies specifically to the host. All service instances, and all service entries for such service instances, depend on the host. When the lease on a host expires, the host and all services that reference it **MUST** be removed at the same time—it is never valid for a service instance to remain when the host it references has been removed. If the KEY record for the host is to remain, the KEY record for any services that reference it **MUST** also remain. However, the service PTR record **MUST** be removed, since it has no key associated with it, and since it is never valid to have a service PTR record for which there is no service instance on the target of the PTR record.

SRP Servers **SHOULD** also track a lease time per service instance. The reason for doing this is that a client may re-register a host with a different set of services, and not remember that some different service instance had previously been registered. In this case, when that service instance lease expires, the SRP server **SHOULD** remove the service instance (although the KEY record for the service instance **SHOULD** be retained until the key lease on that service expires).



This is beneficial because if the SRP client continues to renew the host, but never mentions the stale service again, the stale service will continue to be advertised.

The SRP server MUST include an EDNS(0) Update Lease option in the response if the lease time proposed by the service has been shortened or lengthened. The service MUST check for the EDNS(0) Update Lease option in the response and MUST use the lease times from that option in place of the options that it sent to the server when deciding when to update its registration. The times may be shorter or longer than those specified in the SRP Update; the SRP client must honor them in either case.

SRP clients should assume that each lease ends N seconds after the update was first transmitted, where N is the lease duration. Servers should assume that each lease ends N seconds after the update that was successfully processed was received. Because the server will always receive the update after the SRP client sent it, this avoids the possibility of misunderstandings.

SRP servers MUST reject updates that do not include an EDNS(0) Update Lease option. Dual-use servers MAY accept updates that don't include leases, but SHOULD differentiate between SRP Updates and other updates, and MUST reject updates that would otherwise be SRP Updates if they do not include leases.

Lease times have a completely different function than TTLs. On an authoritative DNS server, the TTL on a resource record is a constant: whenever that RR is served in a DNS response, the TTL value sent in the answer is the same. The lease time is never sent as a TTL; its sole purpose is to determine when the authoritative DNS server will delete stale records. It is not an error to send a DNS response with a TTL of 'n' when the remaining time on the lease is less than 'n'.

## 5. Security Considerations

### 5.1. Source Validation

SRP Updates have no authorization semantics other than first-come, first-served. This means that if an attacker from outside of the administrative domain of the server knows the server's IP address, it can in principle send updates to the server that will be processed successfully. Servers should therefore be configured to reject updates from source addresses outside of the administrative domain of the server.

For updates sent to an anycast IP address of an SRP server, this validation must be enforced by every router on the path from the Constrained-Device Network to the unconstrained portion of the network. For TCP updates, the initial SYN-SYN+ACK handshake prevents updates being forged by an off-network attacker. In order to ensure that this handshake happens, SRP servers relying on three-way-handshake validation MUST NOT accept TCP Fast Open payloads. If the network infrastructure allows it, an SRP server MAY accept TCP Fast Open payloads if all such packets are validated along the path, and the network is able to reject this type of spoofing at all ingress points.

Note that these rules only apply to the validation of SRP Updates. A server that accepts updates from SRP clients may also accept other DNS updates, and those DNS updates may be validated using different rules. However, in the case of a DNS service that accepts SRP updates, the intersection of the SRP Update rules and whatever other update rules are present must be considered very carefully.

For example, a normal, authenticated DNS update to any RR that was added using SRP, but that is authenticated using a different key, could be used to override a promise made by the SRP Server to an SRP client, by replacing all or part of the service registration information with information provided by an authenticated DNS update client. An implementation that allows both kinds of updates should not allow DNS Update clients that are using different authentication and authorization credentials to update records added by SRP clients.

## 5.2. SRP Server Authentication

This specification does not provide a mechanism for validating responses from DNS servers to SRP clients. In the case of Constrained Network/Constrained Node clients, such validation isn't practical because there's no way to establish trust. In principle, a KEY RR could be used by a non-constrained SRP client to validate responses from the server, but this is not required, nor do we specify a mechanism for determining which key to use.

## 5.3. Required Signature Algorithm

For validation, SRP servers MUST implement the ECDSA<sub>P256</sub>SHA256 signature algorithm. SRP servers SHOULD implement the algorithms specified in [RFC8624], Section 3.1, in the validation column of the table, that are numbered 13 or higher and have a "MUST", "RECOMMENDED", or "MAY" designation in the validation column of the table. SRP clients MUST NOT assume that any algorithm numbered lower than 13 is available for use in validating SIG(0) signatures.

## 6. Privacy Considerations

Because DNSSD SRP Updates can be sent off-link, the privacy implications of SRP are different than for multicast DNS responses. Host implementations that are using TCP SHOULD also use TLS if available. Server implementations MUST offer TLS support. The use of TLS with DNS is described in [RFC7858] and [RFC8310].

Hosts that implement TLS support SHOULD NOT fall back to TCP; since servers are required to support TLS, it is entirely up to the host implementation whether to use it.

Public keys can be used as identifiers to track hosts. SRP servers MAY elect not to return KEY records for queries for SRP registrations.

## 7. Delegation of 'service.arpa.'

In order to be fully functional, the owner of the 'arpa.' zone must add a delegation of 'service.arpa.' in the '.arpa.' zone [RFC3172]. This delegation should be set up as was done for 'home.arpa', as a result of the specification in Section 7 of [RFC8375].

## 8. IANA Considerations

### 8.1. Registration and Delegation of 'service.arpa' as a Special-Use Domain Name

IANA is requested to record the domain name 'service.arpa.' in the Special-Use Domain Names registry [SUDN]. IANA is requested, with the approval of IAB, to implement the delegation requested in Section 7.

IANA is further requested to add a new entry to the "Transport-Independent Locally-Served Zones" subregistry of the the "Locally-Served DNS Zones" registry [LSDZ]. The entry will be for the domain 'service.arpa.' with the description "DNS-SD Registration Protocol Special-Use Domain", listing this document as the reference.

### 8.2. 'dnssd-srp' Service Name

IANA is also requested to add a new entry to the Service Names and Port Numbers registry for dnssd-srp with a transport type of tcp. No port number is to be assigned. The reference should be to this document, and the Assignee and Contact information should reference the authors of this document. The Description should be as follows:

Availability of DNS Service Discovery Service Registration Protocol Service for a given domain is advertised using the "\_dnssd-srp.\_tcp.<domain>" SRV record gives the target host and port where DNSSD Service Registration Service is provided for the named domain.

### 8.3. 'dnssd-srp-tls' Service Name

IANA is also requested to add a new entry to the Service Names and Port Numbers registry for dnssd-srp with a transport type of tcp. No port number is to be assigned. The reference should be to this document, and the Assignee and Contact information should reference the authors of this document. The Description should be as follows:

Availability of DNS Service Discovery Service Registration Protocol Service for a given domain over TLS is advertised using the "\_dnssd-srp-tls.\_tcp.<domain>." SRV record gives the target host and port where DNSSD Service Registration Service is provided for the named domain.

### 8.4. Anycast Address

IANA is requested to allocate an IPv6 Anycast address from the IPv6 Special-Purpose Address Registry, similar to the Port Control Protocol anycast address, 2001:1::1. The value TBD should be replaced with the actual allocation in the table that follows. The values for the registry are:

Attribute	value
Address Block	2001:1::TBD/128
Name	DNS-SD Service Registration Protocol Anycast Address
RFC	[this document]
Allocation Date	[date of allocation]
Termination Date	N/A
Source	True
Destination	True
Forwardable	True
Global	True
Reserved-by-protocol	False

Table 1

## 9. Implementation Status

[Note to the RFC Editor: please remove this section prior to publication.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation

and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

There are two known independent implementations of SRP clients:

- \* SRP Client for OpenThread:  
<https://github.com/openthread/openthread/pull/6038>
- \* mDNSResponder open source project: <https://github.com/Abhayakara/mdnsresponder>

There are two related implementations of an SRP server. One acts as a DNS Update proxy, taking an SRP Update and applying it to the specified DNS zone using DNS update. The other acts as an Advertising Proxy [I-D.sctl-advertising-proxy]. Both are included in the mDNSResponder open source project mentioned above.

## 10. Acknowledgments

Thanks to Toke Høiland-Jørgensen, Jonathan Hui, Esko Dijk, Kangping Dong and Abtin Keshavarzian for their thorough technical reviews. Thanks to Kangping and Abtin as well for testing the document by doing an independent implementation. Thanks to Tamara Kemper for doing a nice developmental edit, Tim Wattenberg for doing a SRP client proof-of-concept implementation at the Montreal Hackathon at IETF 102, and Tom Pusateri for reviewing during the hackathon and afterwards.

## 11. Normative References

- [I-D.sekar-dns-ul]  
Cheshire, S. and T. Lemon, "An EDNS0 option to negotiate Leases on DNS Updates", Work in Progress, Internet-Draft, draft-sekar-dns-ul-03, 27 July 2021,  
<<https://datatracker.ietf.org/doc/html/draft-sekar-dns-ul-03>>.
- [RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997,  
<<https://www.rfc-editor.org/info/rfc2132>>.
- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997,  
<<https://www.rfc-editor.org/info/rfc2136>>.

- [RFC2539] Eastlake 3rd, D., "Storage of Diffie-Hellman Keys in the Domain Name System (DNS)", RFC 2539, DOI 10.17487/RFC2539, March 1999, <<https://www.rfc-editor.org/info/rfc2539>>.
- [RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures ( SIG(0)s )", RFC 2931, DOI 10.17487/RFC2931, September 2000, <<https://www.rfc-editor.org/info/rfc2931>>.
- [RFC3172] Huston, G., Ed., "Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa")", BCP 52, RFC 3172, DOI 10.17487/RFC3172, September 2001, <<https://www.rfc-editor.org/info/rfc3172>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC8106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 8106, DOI 10.17487/RFC8106, March 2017, <<https://www.rfc-editor.org/info/rfc8106>>.
- [RFC8375] Pfister, P. and T. Lemon, "Special-Use Domain 'home.arpa.'", RFC 8375, DOI 10.17487/RFC8375, May 2018, <<https://www.rfc-editor.org/info/rfc8375>>.
- [RFC8624] Wouters, P. and O. Sury, "Algorithm Implementation Requirements and Usage Guidance for DNSSEC", RFC 8624, DOI 10.17487/RFC8624, June 2019, <<https://www.rfc-editor.org/info/rfc8624>>.
- [RFC8765] Pusateri, T. and S. Cheshire, "DNS Push Notifications", RFC 8765, DOI 10.17487/RFC8765, June 2020, <<https://www.rfc-editor.org/info/rfc8765>>.
- [SUDN] "Special-Use Domain Names Registry", July 2012, <<https://www.iana.org/assignments/special-use-domain-names/special-use-domain-names.xhtml>>.
- [LSDZ] "Locally-Served DNS Zones Registry", July 2011, <<https://www.iana.org/assignments/locally-served-dns-zones/locally-served-dns-zones.xhtml>>.

## 12. Informative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <<https://www.rfc-editor.org/info/rfc2181>>.
- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <<https://www.rfc-editor.org/info/rfc2782>>.
- [RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <<https://www.rfc-editor.org/info/rfc3007>>.
- [RFC6760] Cheshire, S. and M. Krochmal, "Requirements for a Protocol to Replace the AppleTalk Name Binding Protocol (NBP)", RFC 6760, DOI 10.17487/RFC6760, February 2013, <<https://www.rfc-editor.org/info/rfc6760>>.
- [RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <<https://www.rfc-editor.org/info/rfc6761>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <<https://www.rfc-editor.org/info/rfc7228>>.
- [RFC8310] Dickinson, S., Gillmor, D., and T. Reddy, "Usage Profiles for DNS over TLS and DNS over DTLS", RFC 8310, DOI 10.17487/RFC8310, March 2018, <<https://www.rfc-editor.org/info/rfc8310>>.



- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.
- [RFC8766] Cheshire, S., "Discovery Proxy for Multicast DNS-Based Service Discovery", RFC 8766, DOI 10.17487/RFC8766, June 2020, <<https://www.rfc-editor.org/info/rfc8766>>.
- [I-D.cheshire-dnssd-roadmap]  
Cheshire, S., "Service Discovery Road Map", Work in Progress, Internet-Draft, draft-cheshire-dnssd-roadmap-03, 23 October 2018, <<https://datatracker.ietf.org/doc/html/draft-cheshire-dnssd-roadmap-03>>.
- [I-D.cheshire-edns0-owner-option]  
Cheshire, S. and M. Krochmal, "EDNS0 OWNER Option", Work in Progress, Internet-Draft, draft-cheshire-edns0-owner-option-01, 3 July 2017, <<https://datatracker.ietf.org/doc/html/draft-cheshire-edns0-owner-option-01>>.
- [I-D.sctl-advertising-proxy]  
Cheshire, S. and T. Lemon, "Advertising Proxy for DNS-SD Service Registration Protocol", Work in Progress, Internet-Draft, draft-sctl-advertising-proxy-02, 12 July 2021, <<https://datatracker.ietf.org/doc/html/draft-sctl-advertising-proxy-02>>.
- [ZC] Cheshire, S. and D.H. Steinberg, "Zero Configuration Networking: The Definitive Guide", O'Reilly Media, Inc. , ISBN 0-596-10100-7, December 2005.

#### Appendix A. Testing using standard RFC2136-compliant servers

It may be useful to set up a DNS server for testing that does not implement SRP. This can be done by configuring the server to listen on the anycast address, or advertising it in the `_dnssd-srp._tcp.<zone>` SRV and `_dnssd-srp-tls._tcp.<zone>` record. It must be configured to be authoritative for "default.service.arpa", and to accept updates from hosts on local networks for names under "default.service.arpa" without authentication, since such servers will not have support for FCFS authentication (Section 2.2.4.1).

A server configured in this way will be able to successfully accept and process SRP Updates from services that send SRP updates. However, no prerequisites will be applied, and this means that the

test server will accept internally inconsistent SRP Updates, and will not stop two SRP Updates, sent by different services, that claim the same name(s), from overwriting each other.

Since SRP Updates are signed with keys, validation of the SIG(0) algorithm used by the client can be done by manually installing the client public key on the DNS server that will be receiving the updates. The key can then be used to authenticate the client, and can be used as a requirement for the update. An example configuration for testing SRP using BIND 9 is given in Appendix C.

#### Appendix B. How to allow services to update standard RFC2136-compliant servers

Ordinarily SRP Updates will fail when sent to an RFC 2136-compliant server that does not implement SRP because the zone being updated is "default.service.arpa", and no DNS server that is not an SRP server should normally be configured to be authoritative for "default.service.arpa". Therefore, a service that sends an SRP Update can tell that the receiving server does not support SRP, but does support RFC2136, because the RCODE will either be NOTZONE, NOTAUTH or REFUSED, or because there is no response to the update request (when using the anycast address)

In this case a service MAY attempt to register itself using regular RFC2136 DNS updates. To do so, it must discover the default registration zone and the DNS server designated to receive updates for that zone, as described earlier, using the `_dns-update._udp` SRV record. It can then make the update using the port and host pointed to by the SRV record, and should use appropriate prerequisites to avoid overwriting competing records. Such updates are out of scope for SRP, and a service that implements SRP MUST first attempt to use SRP to register itself, and should only attempt to use RFC2136 backwards compatibility if that fails. Although the owner name for the SRV record specifies the UDP protocol for updates, it is also possible to use TCP, and TCP should be required to prevent spoofing.

#### Appendix C. Sample BIND9 configuration for default.service.arpa.

```
zone "default.service.arpa." {  
    type master;  
    file "/etc/bind/master/service.db";  
    allow-update { key demo.default.service.arpa.; };  
};
```

Figure 1: Zone Configuration in named.conf

```

$ORIGIN .
$TTL 57600 ; 16 hours
default.service.arpa IN SOA      ns3.default.service.arpa.
                                postmaster.default.service.arpa. (
                                2951053287 ; serial
                                3600      ; refresh (1 hour)
                                1800      ; retry (30 minutes)
                                604800    ; expire (1 week)
                                3600      ; minimum (1 hour)
                                )
                                NS       ns3.default.service.arpa.
                                SRV 0 0 53 ns3.default.service.arpa.
$ORIGIN default.service.arpa.
$TTL 3600 ; 1 hour
_ipps._tcp PTR demo._ipps._tcp
$ORIGIN _ipps._tcp.default.service.arpa.
demo TXT "0"
SRV 0 0 9992 demo.default.service.arpa.
$ORIGIN _udp.default.service.arpa.
$TTL 3600 ; 1 hour
_dns-update PTR ns3.default.service.arpa.
$ORIGIN _tcp.default.service.arpa.
_dnssd-srp PTR ns3.default.service.arpa.
$ORIGIN default.service.arpa.
$TTL 300 ; 5 minutes
ns3 AAAA 2001:db8:0:1::1
$TTL 3600 ; 1 hour
demo AAAA 2001:db8:0:2::1
KEY 513 3 13 (
    qweEmaaQ0FAWok5//ftuQtZgiZoiFSUsm0srWREdywQU
    9dpvtOhrdKWUuPT3uEFF5TZU6B4q1z1I662GdaUwqg==
); alg = ECDSA256SHA256 ; key id = 15008
AAAA ::1

```

Figure 2: Example Zone file

## Authors' Addresses

Ted Lemon  
 Apple Inc.  
 One Apple Park Way  
 Cupertino, California 95014  
 United States of America  
 Email: mellon@fugue.com

Stuart Cheshire  
Apple Inc.  
One Apple Park Way  
Cupertino, California 95014  
United States of America  
Phone: +1 408 974 3207  
Email: cheshire@apple.com