

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 13 January 2022

T. Lemon
S. Cheshire
Apple Inc.
12 July 2021

Service Registration Protocol for DNS-Based Service Discovery
draft-ietf-dnssd-srp-10

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 13 January 2022.

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. 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	6
2.2.1. What to publish	7
2.2.2. Where to publish it	7
2.2.3. How to publish it	8
2.2.4. How to secure it	9
2.2.5. Service Behavior	9
2.3. SRP Server Behavior	12
2.3.1. Validation of Adds and Deletes	12
2.3.2. Valid SRP Update Requirements	14
2.3.3. FCFS Name And Signature Validation	15
2.3.4. SRP Update response	16
2.3.5. Optional Behavior	16
3. TTL Consistency	17
4. Maintenance	17
4.1. Cleaning up stale data	17
5. Sleep Proxy	19
6. Security Considerations	20
6.1. Source Validation	20
6.2. SRP Server Authentication	21
6.3. Required Signature Algorithm	21
7. Privacy Considerations	21
8. Delegation of 'service.arpa.'	21
9. IANA Considerations	22
9.1. Registration and Delegation of 'service.arpa' as a Special-Use Domain Name	22
9.2. 'dnssd-srp' Service Name	22
9.3. 'dnssd-srp-tls' Service Name	22
9.4. Anycast Address	23
10. Implementation Status	23
11. Acknowledgments	24
12. Normative References	24
13. 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.

RFC 6763 describes the details of what each of these types of updates contains and is 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.
- * 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. 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 update and each Service Description update. 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. SRP Server Behavior

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 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 update, a Service Description update, or a Host Description update. 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,
- * which points to a Service Instance Name
- * for which a Service Description Instruction is present in the SRP Update
- * if the Service Discovery update 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 an "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 RRs.

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 updates do not modify any other 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 RRset Add to a Service Name and an RRset Add to a Service Instance Name, where 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 same 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) on the public key in the KEY record of the Host Description update. 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 update are processed as if they had been explicitly present: every Service Description that is updated MUST, after the update, 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. 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.

2.3.5. 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 update 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 negotiate longer leases. 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. Sleep Proxy

Another use of SRP is for devices that sleep to reduce power consumption.

In this case, in addition to the DNS Update Lease option [I-D.sekar-dns-ul] described above, the device includes an EDNS(0) OWNER Option [I-D.cheshire-edns0-owner-option].

The EDNS(0) Update Lease option constitutes a promise by the device that it will wake up before this time elapses, to renew its registration and thereby demonstrate that it is still attached to the network. If it fails to renew the registration by this time, that indicates that it is no longer attached to the network, and its registration (except for the KEY in the Host Description) should be deleted.

The EDNS(0) OWNER Option indicates that the device will be asleep, and will not be receptive to normal network traffic. When a DNS server receives a DNS Update with an EDNS(0) OWNER Option, that signifies that the SRP server should set up a proxy for any IPv4 or IPv6 address records in the DNS Update message. This proxy should send ARP or ND messages claiming ownership of the IPv4 and/or IPv6 addresses in the records in question. In addition, the proxy should answer future ARP or ND requests for those IPv4 and/or IPv6 addresses, claiming ownership of them. When the DNS server receives a TCP SYN or UDP packet addressed to one of the IPv4 or IPv6 addresses for which it proxying, it should then wake up the sleeping device using the information in the EDNS(0) OWNER Option. At present version 0 of the OWNER Option specifies the "Wake-on-LAN Magic Packet" that needs to be sent; future versions could be extended to specify other wakeup mechanisms.

Note that although the authoritative DNS server that implements the SRP function need not be on the same link as the sleeping host, the Sleep Proxy must be on the same link.

It is not required that sleepy nodes on a Constrained-Node Network support sleep proxy. Such devices may have different mechanisms for dealing with sleep and wakeup. An SRP registration for such a device will be useful regardless of the mechanism whereby messages are delivered to the sleepy end device. For example, the message might be held in a buffer for an extended period of time by an intermediate device on a mesh network, and then delivered to the device when it wakes up. The exact details of such behaviors are out of scope for this document.

6. Security Considerations

6.1. 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, Service Discovery Protocol 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 registration

protocol, by replacing all or part of the service registration information with information provided by an SRP 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.

6.2. 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.

6.3. Required Signature Algorithm

For validation, SRP servers MUST implement the ECDSA^{P256}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.

7. 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.

8. 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].

9. IANA Considerations

9.1. 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 8.

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.

9.2. 'dnsssd-srp' Service Name

IANA is also requested to add a new entry to the Service Names and Port Numbers registry for dnsssd-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 "_dnsssd-srp._tcp.<domain>." SRV record gives the target host and port where DNSSD Service Registration Service is provided for the named domain.

9.3. 'dnsssd-srp-tls' Service Name

IANA is also requested to add a new entry to the Service Names and Port Numbers registry for dnsssd-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 "_dnsssd-srp-tls._tcp.<domain>." SRV record gives the target host and port where DNSSD Service Registration Service is provided for the named domain.

9.4. 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

10. 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.

11. 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.

12. Normative References

- [I-D.sekar-dns-ul]
Cheshire, S. and T. Lemon, "Dynamic DNS Update Leases", Work in Progress, Internet-Draft, draft-sekar-dns-ul-02, 2 August 2018, <<https://datatracker.ietf.org/doc/html/draft-sekar-dns-ul-02>>.
- [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>>.
- [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>>.

13. 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-01, 22 February 2021, <<https://datatracker.ietf.org/doc/html/draft-sctl-advertising-proxy-01>>.
- [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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 25 April 2022

T. Lemon
S. Cheshire
Apple Inc.
22 October 2021

Service Registration Protocol for DNS-Based Service Discovery
draft-ietf-dnssd-srp-12

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 25 April 2022.

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. 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_{P256}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

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_{P256}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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 24 April 2022

S. Cheshire
Apple Inc.
T. Lemon
Apple Inc
21 October 2021

An EDNS0 option to negotiate Leases on DNS Updates
draft-ietf-dnssd-update-lease-00

Abstract

This document proposes a new EDNS0 option that can be used by DNS Update clients and DNS servers to include a lease lifetime in a DNS Update or response, allowing a server to garbage collect stale resource records that have been added by DNS Updates

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 24 April 2022.

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	2
2. Conventions and Terminology Used in this Document	3
3. Mechanisms	3
4. Update Message Format	3
5. Refresh Messages	4
5.1. Coalescing Refresh Messages	4
5.2. Refresh Message Format	5
5.3. Server Behavior	5
6. Garbage Collection	5
7. Security Considerations	5
8. IANA Considerations	6
9. Acknowledgments	6
10. Normative References	6
11. Informative References	6
Authors' Addresses	7

1. Introduction

Dynamic DNS Update [RFC2136] allows for a mapping from a persistent hostname to a dynamic IP address. This capability is particularly beneficial to mobile hosts, whose IP address may frequently change with location. However, the mobile nature of such hosts often means that dynamically updated resource records are not properly deleted. Consider, for instance, a mobile user who publishes address records via dynamic update. If this user moves their laptop out of range of the Wi-Fi access point, the address record containing stale information may remain on the server indefinitely. An extension to Dynamic Update is thus required to tell the server to automatically delete resource records if they are not refreshed after a period of time.

Note that overloading the resource record TTL [RFC1035] is not appropriate for purposes of garbage collection. Data that is susceptible to frequent change or invalidation, thus requiring a garbage collection mechanism, needs a relatively short resource record TTL to avoid polluting intermediate DNS caches with stale data. Using this TTL, short enough to minimize stale cached data, as a garbage collection lease lifetime would result in an unacceptable amount of network traffic due to refreshes (see Section 5 "Refresh Messages").

2. Conventions and Terminology Used in this Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels", when, and only when, they appear in all capitals, as shown here [RFC2119] [RFC8174].

3. Mechanisms

The EDNS0 Update Lease option is included in a standard DNS Update message [RFC2136] within an EDNS(0) OPT pseudo-RR [RFC6891] with a new OPT and RDATA format proposed here. Encoding the Update Lease Lifetime in an OPT RR requires minimal modification to a name server's front-end, and will cause servers that do not implement this extension to automatically return a descriptive error (NOTIMPL).

4. Update Message Format

Dynamic DNS Update Leases Requests and Responses are formatted as standard DNS Dynamic Update messages [RFC2136], with the addition of a single OPT RR in the Additional section. Note that if a TSIG resource record is to be added to authenticate the update [RFC2845], the TSIG RR should appear *after* the OPT RR, allowing the message digest in the TSIG to cover the OPT RR.

The OPT RR is formatted as follows:

Field Name	Field Type	Description
NAME	domain name	empty (root domain)
TYPE	u_int16_t	OPT
CLASS	u_int16_t	0
TTL	u_int32_t	0
RDLEN	u_int16_t	describes RDATA
RDATA	byte stream	(see below)

RDATA Format:

Field Name	Field Type	Description
OPTION-CODE	u_int16_t	UPDATE-LEASE (2)
OPTION-LENGTH	u_int16_t	4 or 8
LEASE	u_int32_t	desired lease (request) or granted lease (response), in seconds
KEY-LEASE	u_int32_t	optional desired (or granted) lease for KEY records, in seconds

Figure 1

Update Requests contain, in the LEASE field of the OPT RDATA, an unsigned 32-bit integer indicating the lease lifetime, in seconds, desired by the client, represented in network (big-endian) byte order. In Update Responses, this field contains the actual lease granted by the server. The lease granted by the server may be less than, greater than, or equal to the value requested by the client. To reduce network and server load, a minimum lease of 30 minutes (1800 seconds) is RECOMMENDED. Leases are expected to be sufficiently long as to make timer discrepancies (due to transmission latency, etc.) between a client and server negligible. Clients that expect the updated records to be relatively static MAY request appropriately longer leases. Servers MAY grant relatively longer or shorter leases to reduce network traffic due to refreshes, or reduce stale data, respectively.

There are two variants of the EDNS(0) UPDATE-LEASE option, the basic (4-byte) variant and the extended (8-byte) variant.

In the basic (4-byte) variant, the LEASE indicated in the OPT RR applies to all resource records in the Update section.

In the extended (8-byte) variant, the Update Lease communicates two lease lifetimes. The LEASE indicated in the OPT RR applies to all resource records in the Update section *except* for KEY records. The KEY-LEASE indicated in the OPT RR applies to KEY records in the Update section. This variant is used specifically for supporting the DNS-SD Service Registration Protocol [I-D.ietf-dnssd-srp].

5. Refresh Messages

Resource records not to be deleted by the server MUST be refreshed by the client before the lease elapses. Clients SHOULD refresh resource records after 75% of the original lease has elapsed. If the client uses UDP and does not receive a response from the server, the client SHOULD re-try after 2 seconds. The client SHOULD continue to re-try, doubling the length of time between each re-try, or re-try using TCP.

5.1. Coalescing Refresh Messages

If the client has sent multiple updates to a single server, the client MAY include refreshes for all valid updates to that server in a single message. This effectively places all records for a client on the same expiration schedule, reducing network traffic due to refreshes. In doing so, the client includes in the refresh message all existing updates to the server, including those not yet close to expiration, so long as at least one resource record in the message

has elapsed at least 75% of its original lease. If the client uses UDP, the client MUST NOT coalesce refresh messages if doing so would cause truncation of the message; in this case, multiple messages or TCP should be used.

5.2. Refresh Message Format

Refresh messages are formatted like Dynamic Update Leases Requests and Responses (see Section 4 "Update Message Format"). The resource records to be refreshed are contained in the Update section. These same resource records are repeated in the Prerequisite section, as an "RRSet exists (value dependent)" prerequisite [RFC2136]. An OPT RR is the last resource record in the Additional section (except for a TSIG record, which, if required, follows the OPT RR). The OPT RR contains the desired new lease on Requests, and the actual granted lease on Responses. The Update Lease indicated in the OPT RR applies to all resource records in the Update section.

5.3. Server Behavior

Upon receiving a valid Refresh Request, the server MUST send an acknowledgment. This acknowledgment is identical to the Update Response format described in Section 4 "Update Message Format", and contains the new lease of the resource records being refreshed. If no records in the Refresh Request have completed 50% of their leases, the server SHOULD NOT refresh the records; the response should contain the smallest remaining (unrefreshed) lease of all records in the refresh message. The server MUST NOT increment the SOA serial number of a zone as the result of a refresh.

6. Garbage Collection

If the Update Lease of a resource record elapses without being refreshed, the server MUST NOT return the expired record in answers to queries. The server MAY delete the record from its database.

7. Security Considerations

When DNS Update is enabled on an authoritative server, the Security Considerations of that specification [RFC2136] should be considered.

The addition of a record lifetime to facilitate automated garbage collection does not itself add any significant new security concerns.

8. IANA Considerations

The EDNS(0) OPTION CODE 2 has already been assigned for this DNS extension. No additional IANA services are required by this document.

9. Acknowledgments

Thanks to Marc Krochmal and Kiren Sekar to their work in 2006 on the precursor to this document. Thanks also to Roger Pantos and Chris Sharp for their contributions.

10. 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>>.
- [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>>.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<https://www.rfc-editor.org/info/rfc6891>>.
- [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>>.

11. Informative References

- [RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <<https://www.rfc-editor.org/info/rfc2845>>.

[I-D.ietf-dnssd-srp]

Lemon, T. and S. Cheshire, "Service Registration Protocol
for DNS-Based Service Discovery", Work in Progress,
Internet-Draft, draft-ietf-dnssd-srp-10, 12 July 2021,
<<https://datatracker.ietf.org/doc/html/draft-ietf-dnssd-srp-10>>.

Authors' Addresses

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

Phone: +1 408 974 3207
Email: cheshire@apple.com

Ted Lemon
Apple Inc
P.O. Box 958
Brattleboro, Vermont 05302
United States of America

Email: mellon@fugue.com

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

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

An EDNS0 option to negotiate Leases on DNS Updates
draft-ietf-dnssd-update-lease-01

Abstract

This document proposes a new EDNS0 option that can be used by DNS Update clients and DNS servers to include a lease lifetime in a DNS Update or response, allowing a server to garbage collect stale resource records that have been added by DNS Updates

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	2
2. Conventions and Terminology Used in this Document	3
3. Mechanisms	3
4. Update Message Format	3
5. Refresh Messages	4
5.1. Coalescing Refresh Messages	4
5.2. Refresh Message Format	5
5.3. Server Behavior	5
6. Garbage Collection	5
7. Security Considerations	5
8. IANA Considerations	6
9. Acknowledgments	6
10. Normative References	6
11. Informative References	6
Authors' Addresses	7

1. Introduction

Dynamic DNS Update [RFC2136] allows for a mapping from a persistent hostname to a dynamic IP address. This capability is particularly beneficial to mobile hosts, whose IP address may frequently change with location. However, the mobile nature of such hosts often means that dynamically updated resource records are not properly deleted. Consider, for instance, a mobile user who publishes address records via dynamic update. If this user moves their laptop out of range of the Wi-Fi access point, the address record containing stale information may remain on the server indefinitely. An extension to Dynamic Update is thus required to tell the server to automatically delete resource records if they are not refreshed after a period of time.

Note that overloading the resource record TTL [RFC1035] is not appropriate for purposes of garbage collection. Data that is susceptible to frequent change or invalidation, thus requiring a garbage collection mechanism, needs a relatively short resource record TTL to avoid polluting intermediate DNS caches with stale data. Using this TTL, short enough to minimize stale cached data, as a garbage collection lease lifetime would result in an unacceptable amount of network traffic due to refreshes (see Section 5 "Refresh Messages").

2. Conventions and Terminology Used in this Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels", when, and only when, they appear in all capitals, as shown here [RFC2119] [RFC8174].

3. Mechanisms

The EDNS0 Update Lease option is included in a standard DNS Update message [RFC2136] within an EDNS(0) OPT pseudo-RR [RFC6891] with a new OPT and RDATA format proposed here. Encoding the Update Lease Lifetime in an OPT RR requires minimal modification to a name server's front-end, and will cause servers that do not implement this extension to automatically return a descriptive error (NOTIMPL).

4. Update Message Format

Dynamic DNS Update Leases Requests and Responses are formatted as standard DNS Dynamic Update messages [RFC2136], with the addition of a single OPT RR in the Additional section. Note that if a TSIG resource record is to be added to authenticate the update [RFC2845], the TSIG RR should appear *after* the OPT RR, allowing the message digest in the TSIG to cover the OPT RR.

The OPT RR is formatted as follows:

Field Name	Field Type	Description
NAME	domain name	empty (root domain)
TYPE	u_int16_t	OPT
CLASS	u_int16_t	0
TTL	u_int32_t	0
RDLEN	u_int16_t	describes RDATA
RDATA	byte stream	(see below)

RDATA Format:

Field Name	Field Type	Description
OPTION-CODE	u_int16_t	UPDATE-LEASE (2)
OPTION-LENGTH	u_int16_t	4 or 8
LEASE	u_int32_t	desired lease (request) or granted lease (response), in seconds
KEY-LEASE	u_int32_t	optional desired (or granted) lease for KEY records, in seconds

Figure 1

Update Requests contain, in the LEASE field of the OPT RDATA, an unsigned 32-bit integer indicating the lease lifetime, in seconds, desired by the client, represented in network (big-endian) byte order. In Update Responses, this field contains the actual lease granted by the server. The lease granted by the server may be less than, greater than, or equal to the value requested by the client. To reduce network and server load, a minimum lease of 30 minutes (1800 seconds) is RECOMMENDED. Leases are expected to be sufficiently long as to make timer discrepancies (due to transmission latency, etc.) between a client and server negligible. Clients that expect the updated records to be relatively static MAY request appropriately longer leases. Servers MAY grant relatively longer or shorter leases to reduce network traffic due to refreshes, or reduce stale data, respectively.

There are two variants of the EDNS(0) UPDATE-LEASE option, the basic (4-byte) variant and the extended (8-byte) variant.

In the basic (4-byte) variant, the LEASE indicated in the OPT RR applies to all resource records in the Update section.

In the extended (8-byte) variant, the Update Lease communicates two lease lifetimes. The LEASE indicated in the OPT RR applies to all resource records in the Update section *except* for KEY records. The KEY-LEASE indicated in the OPT RR applies to KEY records in the Update section. This variant is used specifically for supporting the DNS-SD Service Registration Protocol [I-D.ietf-dnssd-srp].

5. Refresh Messages

Resource records not to be deleted by the server MUST be refreshed by the client before the lease elapses. Clients SHOULD refresh resource records after 75% of the original lease has elapsed. If the client uses UDP and does not receive a response from the server, the client SHOULD re-try after 2 seconds. The client SHOULD continue to re-try, doubling the length of time between each re-try, or re-try using TCP.

5.1. Coalescing Refresh Messages

If the client has sent multiple updates to a single server, the client MAY include refreshes for all valid updates to that server in a single message. This effectively places all records for a client on the same expiration schedule, reducing network traffic due to refreshes. In doing so, the client includes in the refresh message all existing updates to the server, including those not yet close to expiration, so long as at least one resource record in the message

has elapsed at least 75% of its original lease. If the client uses UDP, the client MUST NOT coalesce refresh messages if doing so would cause truncation of the message; in this case, multiple messages or TCP should be used.

5.2. Refresh Message Format

Refresh messages are formatted like Dynamic Update Leases Requests and Responses (see Section 4 "Update Message Format"). The resource records to be refreshed are contained in the Update section. These same resource records are repeated in the Prerequisite section, as an "RRSet exists (value dependent)" prerequisite [RFC2136]. An OPT RR is the last resource record in the Additional section (except for a TSIG record, which, if required, follows the OPT RR). The OPT RR contains the desired new lease on Requests, and the actual granted lease on Responses. The Update Lease indicated in the OPT RR applies to all resource records in the Update section.

5.3. Server Behavior

Upon receiving a valid Refresh Request, the server MUST send an acknowledgment. This acknowledgment is identical to the Update Response format described in Section 4 "Update Message Format", and contains the new lease of the resource records being refreshed. If no records in the Refresh Request have completed 50% of their leases, the server SHOULD NOT refresh the records; the response should contain the smallest remaining (unrefreshed) lease of all records in the refresh message. The server MUST NOT increment the SOA serial number of a zone as the result of a refresh.

6. Garbage Collection

If the Update Lease of a resource record elapses without being refreshed, the server MUST NOT return the expired record in answers to queries. The server MAY delete the record from its database.

7. Security Considerations

When DNS Update is enabled on an authoritative server, the Security Considerations of that specification [RFC2136] should be considered.

The addition of a record lifetime to facilitate automated garbage collection does not itself add any significant new security concerns.

8. IANA Considerations

The EDNS(0) OPTION CODE 2 has already been assigned for this DNS extension. No additional IANA services are required by this document.

9. Acknowledgments

Thanks to Marc Krochmal and Kiren Sekar to their work in 2006 on the precursor to this document. Thanks also to Roger Pantos and Chris Sharp for their contributions.

10. 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>>.
- [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>>.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<https://www.rfc-editor.org/info/rfc6891>>.
- [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>>.

11. Informative References

- [RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <<https://www.rfc-editor.org/info/rfc2845>>.

[I-D.ietf-dnssd-srp]

Lemon, T. and S. Cheshire, "Service Registration Protocol
for DNS-Based Service Discovery", Work in Progress,
Internet-Draft, draft-ietf-dnssd-srp-12, 24 October 2021,
<<https://datatracker.ietf.org/doc/html/draft-ietf-dnssd-srp-12>>.

Authors' Addresses

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

Ted Lemon
Apple Inc
P.O. Box 958
Brattleboro, Vermont 05302
United States of America
Email: mellon@fugue.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 27 January 2022

T. Lemon
Apple Inc.
26 July 2021

Automatic Replication of DNS-SD Service Registration Protocol Zones
draft-lemon-srp-replication-00

Abstract

This document describes a protocol that can be used for ad-hoc replication of a DNS zone by multiple servers where a single primary DNS authoritative server is not available and the use of stable storage is not desirable.

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 27 January 2022.

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
1.1. Alternatives for maintaining SRP state	3
1.1.1. Primary authoritative DNS service	3
1.1.2. Multicast DNS Advertising Proxy	4
1.1.3. SRP Replication	4
1.2. Implementation	5
1.2.1. Naming of a common service zone	5
1.2.2. Advertising one's own replication service	7
1.2.3. Discovering other replication services	8
1.2.4. Discovering the addresses of peers	9
1.2.5. Establishing Communication with a replication peer	9
1.2.6. Incoming connections	10
1.2.7. Eliminating extra connections	10
1.2.8. Initial synchronization	10
1.2.9. Routine Operation	12
2. Protocol Details	12
2.1. DNS Stateful Operations considerations	12
2.1.1. DSO Session Establishment	12
2.1.2. DSO Session maintenance	13
2.2. DSO Primary TLVs	13
2.2.1. SRPL Session	13
2.2.2. SRPL Send Candidates	14
2.2.3. SRPL Candidate	15
2.2.4. SRPL Host	16
2.3. DSO Secondary TLVs	17
2.3.1. SRPL Candidate Yes	17
2.3.2. SRPL Candidate No	17
2.3.3. SRPL Conflict	18
2.3.4. SRPL Hostname	18
2.3.5. SRPL Host Message	18
2.3.6. SRPL Time Offset	19
2.3.7. SRPL Key ID	19
3. Security Considerations	19
4. Delegation of 'local.arpa.'	19
5. IANA Considerations	19
5.1. 'srpl-tls' Service Name	19
5.2. DSO TLV type code	20
5.3. Registration and Delegation of 'local.arpa' as a Special-Use Domain Name	21
6. Informative References	21
7. Normative References	21
Author's Address	21

1. Introduction

The DNS-SD Service Registration Protocol provides a way for network services to update a DNS zone with DNS-SD information. SRP uses unicast DNS Updates, rather than multicast DNS, to advertise services. This has several advantages over multicast DNS:

- * Reduces reliance on multicast
- * Reduces traffic to devices providing services, which may be constrained devices operating on battery power
- * Allows the advertisement of services on one network link to consumers of such services on a different network link

1.1. Alternatives for maintaining SRP state

1.1.1. Primary authoritative DNS service

Ideally, SRP updates a primary authoritative DNS server for a particular zone. This DNS server acts as the sole source of truth for names within the DNS zone in which SRP services are published. Redundancy is provided by secondary DNS servers, if needed. However, this approach has some drawbacks.

First, it requires 100% availability on the part of a DNS primary authoritative server for the zone. If the primary server is not available for some period of time, new services appearing on the network cannot be registered until primary authoritative service is restored.

The second drawback is that there is no automatic method for managing DNS authoritative service. This means that such a service requires an operator to set it up. What it means to set up such a service is that the following capabilities are provided:

- * An host must be available to act as a primary authoritative DNS server
- * The zone advertised by that server must be delegated, so that the local resolver can successfully answer queries in that zone
- * The local resolver must be able to provide local browsing domain advertisements [RFC6763 section 11].

1.1.2. Multicast DNS Advertising Proxy

An existing alternative to the use of DNS authoritative services for advertising SRP registrations is the advertising proxy [draft-tlsc-advertising-proxy]. An advertising proxy advertises the contents of the SRP update zone using multicast DNS on links on which the need for such advertisements is anticipated. This works well for stub networks [draft-lemon-stub-networks], where services advertised on the stub network must be visible both on the stub network and on the adjacent infrastructure network, but do not generally need to be discoverable on other networks.

One drawback of the advertising proxy model, however, is that there is no shared database from which to advertise services registered by SRP. As a consequence, some of the guarantees provided by SRP, particularly first come, first served naming [draft-ietf-dnssd-srp]. Because advertising proxies are set up automatically on an ad-hoc basis, coordination between advertising proxies is not present, which means that if two devices claim the same name, but register with different SRP servers, the conflict is not detected until the service is advertised using mDNS. In practice, this results in frequent renaming of services, which means that consumers of services need to carefully follow each service that they use as the name changes over time.

An additional drawback is that, from the perspective of the SRP client, SRP service is not unified: SRP servers tend to come and go, and whenever the SRP service with which a particular client has registered goes offline, the client has to notice that this has happened, discover a new SRP server, and re-register, or else it becomes unreachable.

1.1.3. SRP Replication

This document describes a replication mechanism which eliminates the need for a single authoritative source of truth, as in the Primary Authoritative DNS model, while eliminating the drawbacks of the Advertising Proxy model. SRP Replication servers discover each other automatically. Each replication server maintains a copy of the SRP zone which is kept up to date on a best-effort basis.

SRP Replication has several benefits:

- * As long as one SRP replication peer remains online at all times, SRP state is maintained when individual SRP replication peers go offline
- * Name collisions when SRP clients change servers are avoided

- * SRP service on a stub network can appear as an anycast service, so that SRP clients do not see an apparent change in servers and re-register when the server with which they most recently registered goes offline

1.2. Implementation

SRP Replication relies on the fact that any given client is always registering with exactly one SRP server at any given time. This means that when an SRP server receives an SRP update from a client, it can be sure that no other SRP server has a more recent version of that SRP client's registration. Consequently, that SRP server can behave as if it is the source of truth for that client's registration, and other SRP servers can safely assume that any data they have about the client that is less recent can be replaced with the new registration data.

1.2.1. Naming of a common service zone

In order for SRP replication peers to replicate a zone, they must agree upon a common name for the zone. We will describe two mechanisms for agreeing on a common zone here.

1.2.1.1. Zone name based on network name

Network names aren't guaranteed to be unique, but tend to be unique for any given site. In the case of ad-hoc (permissionless) SRP-based service, such as an advertising proxy or an authoritative service using a locally-served zone [<https://www.iana.org/assignments/locally-served-dns-zones/locally-served-dns-zones.xhtml>], because the DNS zone name isn't required to be globally unique, a zone name based on the network name is an easy solution to generating a unique zone name.

When generating a zone name based on a network name, the zone name could be based on a locally configured global zone name, e.g. 'example.com'. It could be based on a locally-managed locally-served name, e.g. 'home.arpa'. Or it could be based on an unmanaged locally-served name, for which we propose to use the root name 'local.arpa.' For the rest of this section we will assume that the specific setting determines which of these domains will be used, and refer to whichever domain that is as DOMAIN.

For zone names based on the network name, the network type should be used as a differentiator, in case there are two different local network types with the same name. So, for example, 'WiFi.DOMAIN.'

1.2.1.1.1. Zone name based on WiFi SSID

If the zone being represented is a WiFi network, then the zone name for the network should be constructed using the WiFi SSID followed by 'WiFi.DOMAIN'. For example, if the SSID is "Example Home" then the zone name would be 'Example Home.WiFi.DOMAIN.' Note that spaces and special characters are allowed in domain names.

1.2.1.1.2. Zone name based on Thread network name

If the zone being represented is a Thread [Thread] network, then the zone name for the network should be constructed using the Thread network name. For example, if the Thread network name is "openthread" then the zone name would be 'openthread.thread.DOMAIN.'

1.2.1.2. Zone name based on local configuration

The above examples assume that it makes sense for each separate subnet to be its own separate zone. However, since SRP guarantees name uniqueness using the first-come, first-served mechanism, it doesn't rely on mDNS's guarantee of per-link uniqueness. Consequently, it is not required that an SRP zone be constrained to the set of services advertised on a single link. For this reason, when it is possible to know that some set of links are all managed by the same set of SRP replication peers, and a name is known for that set of links, that name can be used. To avoid possible collisions, the subdomain 'srp' is used to indicate that this zone is an SRP zone. So in this case the link name would be the locally-known shared name, followed by 'srp.DOMAIN.'

An example of such a scenario would be Apple's HomeKit, in which all HomeKit accessories, regardless of which home network link they are attached to, all are shared in the same namespace. Suppose the HomeKit home's name is "Example Home". In such a situation, the domain name 'Example Home.srp.DOMAIN' could be used.

1.2.1.3. Zone name based on DNS-SD discovery

Another option for naming the local SRP Replication zone would be to use DNS-SD advertisements. This is particularly useful since each SRP replication peer advertises itself using DNS-SD, so there is a convenient place to put this information. To advertise a zone name based on DNS-SD discovery, the SRP Replication peer should add two fields to the TXT record of the service instance. The first field is the domain field: 'domain=name'. This indicates a proposed SRP replication zone name. The second is the join field. If 'join=yes' then other SRP replication servers are encouraged to use the domain name that appears in the domain field rather than creating a new domain.

1.2.2. Advertising one's own replication service

SRP replication service is advertised using DNS-SD [RFC6763]. The service name is '_srpl-tls._tcp'. Each SRP replication peer should have its own hostname, which when combined with the service instance name and the local DNS-SD domain name will produce a service instance name, for example 'example-host._srpl-tls._tcp.local.' The domain under which the service instance name appears will be 'local' for mDNS, and will be whatever domain is used for service registration in the case of a non-mDNS local DNS-SD service.

SRP replication uses DNS port 853 [RFC7858] and is based on DNS Stateful Operations [RFC8490]. Therefore, the SRV record for the example we've given would be:

```
example-host._srpl-tls._tcp.local. IN SRV 0 0 853 example-  
host.local.
```

The TXT record for SRP replication advertises the domain being replicated, permission to join (if applicable), and the server identifier of the SRP replication peer. The server identifier is a 64-bit number encoded as hexadecimal ASCII, produced with a high-quality random number generator [RFC4086]. This identifier need not be persistent across SRP replication peer restarts. So in our example the TXT record might look like this:

```
#domain=openthread.thread.home.arpa.\032server-id=eb5bb51919a15cec
```

(Note that each name/value pair in the TXT record is length-encoded, so the '#' and the '\032' are the lengths of the two name/value pairs.)

1.2.3. Discovering other replication services

An SRP Replication Peer MUST maintain an ongoing DNS-SD browse on the service name `'_srpl-tls._tcp'` within the local browsing domain. The ongoing browse will produce two different types of events: "add" events and "remove" events. When the browse is started, it should produce an 'add' event for every SRP replication partner currently present on the network, including the peer that is doing the browsing. Whenever a partner goes offline, a 'remove' event should be produced. 'remove' events are not guaranteed, however.

When a new service is added, the SRP peer checks to see if it is in a compatible domain. If the SRP peer has a domain to advertise, it compares that domain to the domain advertised in the added service instance: if they are not the same, then this instance is not a candidate for connection, and should be ignored.

If the SRP peer does not have a domain to advertise, then when it begins to browse for partners, it sets a timer for `DOMAIN_DISCOVERY_TIMEOUT` seconds.

If the SRP peer does not have a domain to advertise, and is therefore willing to join an existing domain, it checks to see if the TXT record for the service indicates that joining is permitted. If so, the SRP peer adopts the provided domain name. Once it has adopted such a domain name, it updates its own TXT record to indicate that domain name, and sets the `'join=yes'` key/value pair in the TXT record. It also cancels the `DOMAIN_DISCOVERY_TIMEOUT` timer.

If the `DOMAIN_DISCOVERY_TIMEOUT` timer goes off, then the SRP peer MUST propose a zone name using one of the methods mentioned previously. It advertises that zone name in its TXT record, with `'join=yes'`. It then sets a new timer for `DOMAIN_PROPOSE_TIMEOUT` seconds.

While waiting for the `DOMAIN_PROPOSE_TIMEOUT` timer to go off, any new 'add' events that arrive are examined to see if they are potential domains to join. If a potential domain to join is seen, and it is the same as the proposed domain, then the peer adopts that domain and treats it as its domain to advertise. It then cancels the `DOMAIN_DISCOVERY_TIMEOUT` timer.

When the `DOMAIN_DISCOVERY_TIMEOUT` timer expires, the peer initializes the domain to be advertised using the one that it chose, and the chosen server-id to be its own. It then iterates across the list of 'add' events that have been seen. Each advertisement is examined, comparing its server-id to the chosen server-id. If the chosen server-id is numerically greater than the server-id in the

advertisement, then the domain to be advertised and the chosen server-id are updated from the advertisement. At the end of this process, the peer adopts whatever domain is now set as the domain to be advertised.

Once a domain has been chosen, a list of partners in that domain can be generated from the list of add events previously seen. When a new add event is seen that advertises the peer's domain to be advertised, that partner is added to the list of partners, if not already present. When a remove event is seen, if that partner is on the list of partners, a timer is set for DOMAIN_INSTANCE_TIMEOUT seconds.

When the timer for DOMAIN_INSTANCE_TIMEOUT timer expires, if the partner that was removed has not been re-added, it is removed from the list of partners and any connection to it is dropped.

1.2.4. Discovering the addresses of peers

When a partner is discovered, two new ongoing mDNS queries are started on the hostname indicated in the SRV record of the partner: one for A records, and one for AAAA records. Each time an address 'add' event is seen, either for an 'A' record or an 'AAAA' record, the peer adds the address to the list of addresses belonging to that partner.

1.2.5. Establishing Communication with a replication peer

When an address is added to a partner's address list, the peer first checks to see if the address is one of its own addresses. If so, then the partner is marked "me", and no connection is attempted to it. This is somewhat safer than comparing hostnames, since a hostname collision can result in renaming.

If the partner is not marked 'me', then the peer checks to see if it has an existing outgoing connection to that partner. If it does not, then it checks to see whether it has disabled outgoing connections to that partner. If not, then it attempts to connect on the new address.

When a connection fails, it advances to the next address in the list, if there is one. If there are no remaining addresses, the peer sets a timer for RECONNECT_INTERVAL seconds. When this timer expires, it starts again at the beginning of the list and attempts to connect to the first address, iterating again across the list until a connection succeeds or it runs out of addresses.

Additionally, when an address is added, it is checked against the list of unidentified incoming connections. If a match is found, and the partner is marked "me," then the unidentified connection is removed from the list and dropped. Otherwise, it is attributed to the matching partner, and the protocol is started at the point of receiving an incoming connection.

When an outgoing connection succeeds, the peer sends its server ID.

1.2.6. Incoming connections

When an incoming connection is received, it is checked against the partner list based on the source address of the incoming connection. If the address appears on the list of addresses for a partner, then the connection is attributed to that partner. If no matching partner is found, a timer of UNIDENTIFIED_PARTNER_TIMEOUT seconds is set, and the incoming connection is added to the list of "unidentified" connections.

If a matching partner is found, then the peer waits for an incoming partner ID. When such an ID is received, it is compared to the peer's server-id. If the incoming server ID is the same as or greater than the peer's server ID, the connection is dropped. Otherwise, the connection proceeds to the "initial synchronization" state.

1.2.7. Eliminating extra connections

When an outgoing connection succeeds, the peer sends its server ID to the partner. When an incoming connection succeeds, the peer waits for a server ID. Because both connections are peer connections, and we only need one connection, the peer with the higher server ID acts as the client and the peer with the lower server ID acts as the server. If the server IDs are equal, then the connecting server generates a new server ID, updates its TXT record, and re-does the comparison.

1.2.8. Initial synchronization

The connecting peer begins the session by sending its server ID. The receiving peer waits for a server ID, and when it receives one, does the server ID comparison mentioned earlier. If the connection survives the comparison, then the server sends a response to the session message and waits for the client to request a list of update candidates.

The connecting peer waits for a response to the initial session message, and when it is received, requests that the server send candidates.

1.2.8.1. Sending candidates

When a peer receives a "send candidates" message that it is expecting to receive, it generates a candidate list from the list of known SRP clients. This list includes SRP clients that have registered directly with the peer, and SRP clients that have been received through SRP replication updates. Each candidate contains a hostname, a time offset, and a key identifier.

The key identifier is computed as follows:

```
uint32_t key_id(uint8_t *key_data, int key_len) {
    uint32_t key_id = 0;
    for (int i = 0; i < key_data_len; i += 4) {
        key_id += ((key_data[i] << 24) | (key_data[i + 1] << 16) |
                  (key_data[i + 2] << 8) | (key_data[i + 3]));
    }
    return key_id;
}
```

When a peer receives a candidate message during the synchronization process, it first searches for an SRP registration with a hostname that matches the hostname in the candidate message. It then compares the key ID to the key ID in the candidate message. If the key ID doesn't match, it sends back a candidate response status of "conflict". If the key ID does match, it compares the time provided to the time the existing host entry was received. If the time of the update is later, it sends a "send host" response. If it is earlier or the same, it sends a "continue" response. If there is no matching host entry for the candidate message, the peer sends a "send host" response.

When a peer receives a candidate response with a status of "send host", it generates a host message, which contains the hostname, the time offset, and the SRP message that was received from the host. The peer then applies the SRP update message as if it had been received directly from the SRP client. The host update time sent by the partner is remembered as the time when the update was received from the client, for the purposes of future synchronization.

When a peer is finished iterating across its list of candidates, it sends a "send candidates" response.

When a peer receives a "send candidates" response, if it is the server, it sends its own "send candidates" message, and processes any proposed candidates.

When a peer that is a server receives a "send candidates" response, it goes into the "routine operation" state. When a peer that is a client sends its "send candidates" response, it goes into the "routine operation" state.

1.2.9. Routine Operation

During routine operation, whenever an update is successfully processed from an SRP client, the peer that received that update queues that update to be sent to each partner to which it has a connection, whether server or client. If there are no updates pending to a particular client, the update is sent immediately. Otherwise, it's sent when the outstanding update is acknowledged.

When during routine operation a peer receives a host update from its partner, it immediately applies that update to its local SRP zone. This is based on the assumption that a new update is always more current than a copy of the host information in its database.

2. Protocol Details

The DNS-SD SRP Replication Protocol (henceforth SRPL) is based on DNS Stateful Operations [RFC8490]. Each SRP replication peer creates a listener on port 853, the DNS-over-TLS [RFC7858] reserved port. This listener can be used for other DNS requests as well.

Participants in the protocol are peers. To distinguish between peers, the terms "peer" and "partner" are used. "Peer" refers to the peer that is communicating or receiving communication. "Partner" refers to the other peer. Peers can be clients or servers: a peer that has established a connection to a partner is a client; a peer that has received a connection from a partner is a server.

2.1. DNS Stateful Operations considerations

DNS Stateful Operations is a DNS per-connection session management protocol. DNS Push session management includes session establishment as well as session maintenance.

2.1.1. DSO Session Establishment

An DSO session for an SRPL connection can be established either by simply sending the first SRPL message, or by sending a DSO Keepalive message. Section 5.1 of [RFC8490].

2.1.2. DSO Session maintenance

DSO sessions can be active or idle. As long as the SRPL protocol is active on a connection, the DSO state of the connection is active. DSO sessions require occasional keepalive messages. The default of fifteen seconds is adequate for SRPL.

An idle DSO session must persist for long enough that there is a chance for the browse that identifies it to succeed. Therefore, the minimum DSO session inactivity timeout is `2*UNIDENTIFIED_PARTNER_TIMEOUT` seconds.

2.2. DSO Primary TLVs

Each DSO message begins with a primary TLV, and contains secondary TLVs with additional information. The primary TLVs used in the SRPL protocol are as follows:

2.2.1. SRPL Session

DSO-TYPE code: `SRPLSession`. Introduces the SRPL session. In addition to the header and length, the SRPL Session message includes a server ID, which is a 64-bit unsigned number in network byte order. The SRPL Session primary TLV does not include any secondary TLVs. SRPL Session requests are DSO requests: the recipient is expected to send a response TLV. Both request and response TLVs have the same format.

2.2.1.1. SRPL client behavior

The SRPL Session request is sent by a peer acting as a client to its partner once the TLS connection to the partner, acting as a server, has succeeded. The SRPL session message establishes the DSO connection as an SRP protocol connection. If it is the first DSO message sent by the peer acting as a client, then it also establishes the DSO session.

When the SRPL peer acting as a client receives a response to its SRPL session message, it sends an SRPL Send Candidates message.

2.2.1.2. SRPL server behavior

An SRPL peer acting as a server that receives an SRPL Session request checks to see if the connection on which it was received is already established. If so, this is a protocol error, and the SRPL peer MUST drop the connection.

It then compares the server ID sent by the partner to its own server ID. If the partner's server ID is numerically less than the server's server ID, the server MUST drop the connection.

If the server ID of the peer is identical to the partner's server ID, then the server generates a new server ID and updates its TXT record with the new server ID.

If the peer acting as a server did not drop the incoming connection, then it sends an SRPL Session response containing its current server ID.

2.2.2. SRPL Send Candidates

DSO-TYPE code: SRPLSendCandidates. Requests the peer to send its candidates list. The SRPL Send Candidates message contains no additional data. The SRPL Send Candidates primary TLV does not include any secondary TLVs. SRPL Send Candidates messages are DSO requests: the recipient is expected to send a response TLV. Both request and response TLVs have the same format.

2.2.2.1. SRPL client behavior

An SRPL peer acting as a client MUST send an SRPL Send Candidates request after it has received an SRPL Session response. It MUST NOT send this request at any other time.

An SRPL peer acting as a client expects to receive an SRPL Send Candidates message after it has received an SRPL Send Candidates response. If it receives an SRPL Send Candidates message at any other time, this is a protocol error, and the SRPL peer should drop its connection to the server.

2.2.2.2. SRPL server behavior

An SRPL peer acting as a server expects to receive an SRPL Send Candidates request after it has sent an SRPL Session response. If it receives an SRPL Candidates request at any other time, this is a protocol error, and it MUST drop the connection.

An SRPL peer acting as a server MUST send an SRPL Send Candidates request after it has sent an SRPL Send Candidates response.

An SRPL peer acting as a server MUST enter the "normal operations" state after receiving an SRPL Send Candidates response from its partner.

2.2.3. SRPL Candidate

DSO-TYPE code: SRPLCandidate. Announces the availability of a specific candidate SRP client registration. The SRPL Candidate message contains no additional data. SRPL Candidate messages are DSO requests: the recipient is expected to send a response TLV. Both request and response TLVs have the same format.

2.2.3.1. Required secondary TLVs

The SRPL Candidate request MUST include the following secondary TLVs: SRPL Hostname, SRPL Time Offset, and SRPL Key ID. If an SRPL peer receives an SRPL Candidate request that doesn't contain all of these secondary TLVs, this is a protocol error, and the peer MUST drop the connection.

The SRPL Candidate response MUST include one of the following status TLVs: SRPL Candidate Yes, SRPL Candidate No, or SRPL Conflict. If an SRPL peer receives an SRPL Candidate response which does not contain exactly one of these TLVs, this is a protocol error, and the peer MUST drop the connection.

2.2.3.2. SRPL peer common behavior

SRPL peers expect to receive SRPL Candidate messages between the time that they have sent an SRPL Send Candidates message and the time that they have received an SRPL Send Candidates response. If an SRPL Candidate message is received at any other time, this is a protocol error, and the peer MUST drop the connection.

Peers MUST NOT send SRPL Candidate requests if they have sent any SRPL Candidate or SRPL host requests that have not yet received responses. Peers receiving SRPL Candidate requests when they have not yet responded to an outstanding SRPL Candidate request or SRPL Host request MUST treat this as a protocol failure and drop the connection.

When a peer receives a valid SRPL Candidate message, it checks its SRP registration database for a host that matches both the SRPL Hostname and SRPL Key ID TLVs. If such a match is not found, the peer sends an SRPL Candidate response that includes the SRPL Candidate Yes secondary TLV.

If a match is found for the hostname, but the Key ID doesn't match, this is a conflict, and the peer sends an SRPL Candidate response with the SRPL Conflict secondary TLV.

If a match is found for the hostname, and the key ID matches, then the peer computes the update time of the candidate by subtracting the value of the SRPL Time Offset TLV from the current time in seconds. This computation should be done when the SRPL Candidate message is received to avoid clock skew. If 'candidate update time' - 'local update time' is greater than SRPL_UPDATE_SKEW_WINDOW, then the candidate update is more recent than the current SRP registration. In this case, the peer sends an SRPL Candidate response and includes the SRPL Candidate Yes secondary TLV. The reason for adding in some skew is to account for network transmission delays.

2.2.4. SRPL Host

DSO-TYPE code: SRPLHost. Provides the content of a particular SRP client registration. The SRPL Host message contains no additional data. SRPL Host messages are DSO requests: the recipient is expected to send a response TLV. Both request and response TLVs have the same format.

2.2.4.1. Required secondary TLVs

The SRPL Host request MUST include the following secondary TLVs: SRPL Hostname, SRPL Time Offset, and SRPL Key ID. If an SRPL peer receives an SRPL Candidate request that doesn't contain all of these secondary TLVs, this is a protocol error, and the peer MUST drop the connection.

2.2.4.2. SRPL peer common behavior during synchronization

SRPL peers expect to receive either zero or one SRPL Host requests after sending an SRPL Candidate response with a SRPL Candidate Yes secondary TLV. If an SRPL Host request is received at any other time during synchronization, this is a protocol error, and the peer MUST drop the connection. The only time that an SRPL Host request would not follow a positive SRPL Candidate response would be when the candidate host entry's lease expired after the SRPL Candidate request was sent but before the SRPL Candidate response was received.

SRPL peers send SRPL Host requests during synchronization when a valid SRPL Candidate response has been received that includes an SRPL Candidate Yes secondary TLV. The host request is generated based on the current candidate (the one for which the SRPL Candidate request being responded to was sent).

2.2.4.3. SRPL peer common behavior during normal operations

When an SRPL peer during normal operations receives and has successfully validated an SRP update from an SRP client, it MUST send that update to each of its connected partners as an SRPL Host request. If the connection to a particular partner is not busy, and there are no updates already queued to be sent, it MUST send the SRPL Host message immediately. Otherwise, it MUST queue the update to send when possible. The queue MUST be first-in, first-out.

After an SRPL peer has sent an SRPL Host request to a partner, and before it receives a corresponding SRPL Host response, it MUST NOT send any more SRPL Host messages to that partner.

When an SRPL peer receives an SRPL Host request during normal operations, it MUST apply it immediately. While it is being applied, it MUST NOT send any other SRPL Host requests to that peer.

When an SRPL Host request has been successfully applied by an SRPL peer, the peer MUST send an SRPL Host response.

If an SRPL peer receives an SRPL Host request while another SRPL Host request is being processed, this is a protocol error, and the peer MUST drop the connection to its partner.

2.3. DSO Secondary TLVs

In addition to the Primary TLVs used to send requests between SRPL peers, we define secondary TLVs to carry formatter information needed for various SRPL requests.

2.3.1. SRPL Candidate Yes

DSO-TYPE code: SRPLCandidateYes. In an SRPL Candidate response, indicates to the partner that an SRPL Host message for the candidate is wanted and should be sent.

Appears as a secondary TLV in SRPL Candidate responses. MUST NOT appear in any other SRPL request or response. MUST NOT appear in addition to either SRPL Conflict or SRPL Candidate No secondary TLVs.

2.3.2. SRPL Candidate No

DSO-TYPE code: SRPLCandidateNo. In an SRPL Candidate response, indicates to the partner that an SRPL Host message for the candidate is not wanted and should not be sent.

Appears as a secondary TLV in SRPL Candidate responses. MUST NOT appear in any other SRPL request or response. MUST NOT appear in addition to either SRPL Conflict or SRPL Candidate Yes secondary TLVs.

2.3.3. SRPL Conflict

DSO-TYPE code: SRPLConflict. In an SRPL Candidate response, indicates to the partner that an SRPL Host message for the candidate is not wanted and should not be sent. Additionally indicates that the proposed host conflicts with local data. This indication is informative and has no effect on processing.

Appears as a secondary TLV in SRPL Candidate responses. MUST NOT appear in any other SRPL request or response. MUST NOT appear in addition to either SRPL Candidate Yes or SRPL Candidate No secondary TLVs.

2.3.4. SRPL Hostname

DSO-TYPE code: SRPLHostname. In an SRPL Candidate or SRPL Host request, indicates to the partner the hostname of an SRP registration.

Required as a secondary TLV in SRPL Candidate and SRPL Host requests. MUST NOT appear in any other SRPL request or response.

2.3.5. SRPL Host Message

DSO-TYPE code: SRPLHostMessage. In an SRPL Host request, conveys the literal contents of the SRP update that resulted in the SRP Host registration being updated. The content of the SRPL Host Message is used to update the host on the peer receiving the request. Note that the SRP message being sent can't be modified by the SRPL peer sending it, so in order to validate the message (assuming that the signature includes a nonzero time), the validation process should adjust the current time by the time offset included in the SRPL Time Offset TLV when comparing against the signature time when checking for replay attacks. The computation of the current time of signing should be done when the message is received to avoid clock skew that might result from processing delays.

Required as a secondary TLV in SRPL Host requests. MUST NOT appear in any other SRPL request or response.

2.3.6. SRPL Time Offset

DSO-TYPE code: SRPLTimeOffset. In an SRPL Candidate or SRPL Host request, conveys the difference between the time the SRP update was received from the SRP client and the current time on the peer generating the request, in seconds.

Required as a secondary TLV in SRPL Candidate and SRPL Host requests. MUST NOT appear in any other SRPL request or response.

2.3.7. SRPL Key ID

DSO-TYPE code: SRPLKeyID. In an SRPL Candidate, conveys the key ID of the SRP client.

Required as a secondary TLV in SRPL Candidate requests. MUST NOT appear in any other SRPL request or response.

3. Security Considerations

SRP replication basically relies on the trustworthiness of hosts on the local network. Since SRP itself relies on the same level of trust, SRP replication doesn't make things worse. However, when the option to have a central SRP server is available, that is likely to be more trustworthy.

4. Delegation of 'local.arpa.'

In order to be fully functional, the owner of the 'arpa.' zone must add a delegation of 'local.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].

5. IANA Considerations

5.1. 'srpl-tls' Service Name

IANA is requested to add a new entry to the Service Names and Port Numbers registry for srpl-tls 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-SD SRP Replication Service for a given domain is advertised using the "_srpl-tls._tcp.<domain>." SRV record gives the target host and port where DNS-SD SRP Replication Service is provided for the named domain.

5.2. DSO TLV type code

The IANA is requested to add the following entries to the 16-bit DSO Type Code Registry. Each type mnemonic should be replaced with an allocated type code, both in this table and elsewhere in the document. RFC-TBD should be replaced with the name of this document once it becomes an RFC.

Type	Name	Early Data	Status	Reference
SRPLSession	SRPL Session	No	STD	RFC-TBD
SRPLSendCandidates	SRPL Send Candidates	No	STD	RFC-TBD
SRPLCandidate	SRPL Candidate	No	STD	RFC-TBD
SRPLHost	SRPL Host	No	STD	RFC-TBD
SRPLCandidateYes	SRPL Candidate Yes	No	STD	RFC-TBD
SRPLCandidateNo	SRPL Candidate No	No	STD	RFC-TBD
SRPLConflict	SRPL Conflict	No	STD	RFC-TBD
SRPLHostname	SRPL Hostname	No	STD	RFC-TBD
SRPLHostMessage	SRPL Host Message	No	STD	RFC-TBD
SRPLTimeOffset	SRPL Time Offset	No	STD	RFC-TBD
SRPLKeyID	SRPL Key ID	No	STD	RFC-TBD

Table 1

5.3. Registration and Delegation of 'local.arpa' as a Special-Use Domain Name

IANA is requested to record the domain name local.arpa.' in the Special-Use Domain Names registry [SUDN]. IANA is requested, with the approval of IAB, to implement the delegation requested in Section 4.

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 local.arpa.' with the description "Ad-hoc DNS-SD Special-Use Domain", listing this document as the reference.

6. Informative References

7. Normative References

- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.

Author's Address

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

Email: mellon@fugue.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 11 May 2022

T. Lemon
Apple Inc.
7 November 2021

Automatic Replication of DNS-SD Service Registration Protocol Zones
draft-lemon-srp-replication-01

Abstract

This document describes a protocol that can be used for ad-hoc replication of a DNS zone by multiple servers where a single primary DNS authoritative server is not available and the use of stable storage is not desirable.

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 11 May 2022.

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
1.1. Alternatives for maintaining SRP state	3
1.1.1. Primary authoritative DNS service	3
1.1.2. Multicast DNS Advertising Proxy	4
1.1.3. SRP Replication	4
2. Implementation	5
2.1. Naming of a common service zone	5
2.1.1. Zone name based on network name	5
2.1.2. Zone name based on local configuration	6
2.1.3. Zone name based on DNS-SD discovery	6
2.2. Advertising one's own replication service	7
2.3. Discovering other replication services	8
2.3.1. Getting an SRP zone name from other SRP services . .	8
2.3.2. Establishing Replication with an existing Dataset . .	9
2.3.3. Establishing Replication When There Is No Existing Dataset	11
2.3.4. Conflicting precedence values	11
2.3.5. Handling primary transitions	11
2.4. Discovering the addresses of partners	12
2.5. Establishing Communication with a replication partner . .	12
2.6. Incoming connections	13
2.7. Eliminating extra connections	13
2.8. Initial synchronization	13
2.8.1. Sending candidates	14
2.9. Routine Operation	15
3. Protocol Details	15
3.1. DNS Stateful Operations considerations	15
3.1.1. DSO Session Establishment	15
3.1.2. DSO Session maintenance	16
3.2. DSO Primary TLVs	16
3.2.1. SRPL Session	16
3.2.2. SRPL Send Candidates	17
3.2.3. SRPL Candidate	18
3.2.4. SRPL Host	19
3.3. DSO Secondary TLVs	20
3.3.1. SRPL Candidate Yes	20
3.3.2. SRPL Candidate No	21
3.3.3. SRPL Conflict	21
3.3.4. SRPL Hostname	21
3.3.5. SRPL Host Message	21
3.3.6. SRPL Time Offset	22
3.3.7. SRPL Key ID	22
4. Security Considerations	22
5. Delegation of 'local.arpa.'	23
6. IANA Considerations	23
6.1. 'srpl-tls' Service Name	23

6.2. DSO TLV type code	23
6.3. Registration and Delegation of 'local.arpa' as a Special-Use Domain Name	24
7. Informative References	25
8. Normative References	25
Author's Address	25

1. Introduction

The DNS-SD Service Registration Protocol provides a way for network services to update a DNS zone with DNS-SD information. SRP uses unicast DNS Updates, rather than multicast DNS, to advertise services. This has several advantages over multicast DNS:

- * Reduces reliance on multicast
- * Reduces traffic to devices providing services, which may be constrained devices operating on battery power
- * Allows the advertisement of services on one network link to consumers of such services on a different network link

1.1. Alternatives for maintaining SRP state

1.1.1. Primary authoritative DNS service

Ideally, SRP updates a primary authoritative DNS server for a particular zone. This DNS server acts as the sole source of truth for names within the DNS zone in which SRP services are published. Redundancy is provided by secondary DNS servers, if needed. However, this approach has some drawbacks.

First, it requires 100% availability on the part of a DNS primary authoritative server for the zone. If the primary server is not available for some period of time, new services appearing on the network cannot be registered until primary authoritative service is restored.

The second drawback is that there is no automatic method for managing DNS authoritative service. This means that such a service requires an operator to set it up. What it means to set up such a service is that the following capabilities are provided:

- * An host must be available to act as a primary authoritative DNS server
- * The zone advertised by that server must be delegated, so that the local resolver can successfully answer queries in that zone

- * The local resolver must be able to provide local browsing domain advertisements [RFC6763 section 11].

1.1.2. Multicast DNS Advertising Proxy

An existing alternative to the use of DNS authoritative services for advertising SRP registrations is the advertising proxy [draft-tlsc-advertising-proxy]. An advertising proxy advertises the contents of the SRP update zone using multicast DNS on links on which the need for such advertisements is anticipated. This works well for stub networks [draft-lemon-stub-networks], where services advertised on the stub network must be visible both on the stub network and on the adjacent infrastructure network, but do not generally need to be discoverable on other networks.

One drawback of the advertising proxy model, however, is that there is no shared database from which to advertise services registered by SRP. As a consequence, some of the guarantees provided by SRP, particularly first come, first served naming [draft-ietf-dnssd-srp]. Because advertising proxies are set up automatically on an ad-hoc basis, coordination between advertising proxies is not present, which means that if two devices claim the same name, but register with different SRP servers, the conflict is not detected until the service is advertised using mDNS. In practice, this results in frequent renaming of services, which means that consumers of services need to carefully follow each service that they use as the name changes over time.

An additional drawback is that, from the perspective of the SRP client, SRP service is not unified: SRP servers tend to come and go, and whenever the SRP service with which a particular client has registered goes offline, the client has to notice that this has happened, discover a new SRP server, and re-register, or else it becomes unreachable.

1.1.3. SRP Replication

This document describes a replication mechanism which eliminates the need for a single authoritative source of truth, as in the Primary Authoritative DNS model, while eliminating the drawbacks of the Advertising Proxy model. SRP Replication servers discover each other automatically. Each replication server maintains a copy of the SRP zone which is kept up to date on a best-effort basis.

SRP Replication has several benefits:

- * As long as one SRP replication partner remains online at all times, SRP state is maintained when individual SRP replication partners go offline
- * Name collisions when SRP clients change servers are avoided
- * SRP service on a stub network can appear as an anycast service, so that SRP clients do not see an apparent change in servers and re-register when the server with which they most recently registered goes offline

2. Implementation

SRP Replication relies on the fact that any given client is always registering with exactly one SRP server at any given time. This means that when an SRP server receives an SRP update from a client, it can be sure that no other SRP server has a more recent version of that SRP client's registration. Consequently, that SRP server can behave as if it is the source of truth for that client's registration, and other SRP servers can safely assume that any data they have about the client that is less recent can be replaced with the new registration data.

2.1. Naming of a common service zone

In order for SRP replication partners to replicate a zone, they must agree upon a common name for the zone. We will describe two mechanisms for agreeing on a common zone here.

2.1.1. Zone name based on network name

Network names aren't guaranteed to be unique, but tend to be unique for any given site. In the case of ad-hoc (permissionless) SRP-based service, such as an advertising proxy or an authoritative service using a locally-served zone [<https://www.iana.org/assignments/locally-served-dns-zones/locally-served-dns-zones.xhtml>], because the DNS zone name isn't required to be globally unique, a zone name based on the network name is an easy solution to generating a unique zone name.

When generating a zone name based on a network name, the zone name could be based on a locally configured global zone name, e.g. 'example.com'. It could be based on a locally-managed locally-served name, e.g. 'home.arpa'. Or it could be based on an unmanaged locally-served name, for which we propose to use the root name 'local.arpa.' For the rest of this section we will assume that the specific setting determines which of these domains will be used, and refer to whichever domain that is as DOMAIN.

For zone names based on the network name, the network type should be used as a differentiator, in case there are two different local network types with the same name. So, for example, 'WiFi.DOMAIN.'

2.1.1.1. Zone name based on WiFi SSID

If the zone being represented is a WiFi network, then the zone name for the network should be constructed using the WiFi SSID followed by 'WiFi.DOMAIN'. For example, if the SSID is "Example Home" then the zone name would be 'Example Home.WiFi.DOMAIN.' Note that spaces and special characters are allowed in domain names.

2.1.1.2. Zone name based on Thread network name

If the zone being represented is a Thread [Thread] network, then the zone name for the network should be constructed using the Thread network name. For example, if the Thread network name is "openthread" then the zone name would be 'openthread.thread.DOMAIN.'

2.1.2. Zone name based on local configuration

The above examples assume that it makes sense for each separate subnet to be its own separate zone. However, since SRP guarantees name uniqueness using the first-come, first-served mechanism, it doesn't rely on mDNS's guarantee of per-link uniqueness. Consequently, it is not required that an SRP zone be constrained to the set of services advertised on a single link. For this reason, when it is possible to know that some set of links are all managed by the same set of SRP replication partners, and a name is known for that set of links, that name can be used. To avoid possible collisions, the subdomain 'srp' is used to indicate that this zone is an SRP zone. So in this case the link name would be the locally-known shared name, followed by 'srp.DOMAIN.'

An example of such a scenario would be Apple's HomeKit, in which all HomeKit accessories, regardless of which home network link they are attached to, all are shared in the same namespace. Suppose the HomeKit home's name is "Example Home". In such a situation, the domain name 'Example Home.srp.DOMAIN' could be used.

2.1.3. Zone name based on DNS-SD discovery

Another option for naming the local SRP Replication zone would be to use DNS-SD advertisements. This is particularly useful since each SRP replication partner advertises itself using DNS-SD, so there is a convenient place to put this information. To advertise a zone name based on DNS-SD discovery, the SRP replication partner should add two fields to the TXT record of the service instance. The first field is

the domain field: 'domain=name'. This indicates a proposed SRP replication zone name. The second is the join field. If 'join=yes' then other SRP replication servers are encouraged to use the domain name that appears in the domain field rather than creating a new domain.

2.2. Advertising one's own replication service

An SRP replication service only advertises its replication service if no other service for the domain it is replicating is already present, or else if it has successfully connected to and synchronized with all of the SRP services it sees advertised for the domain it is configured to replicate.

SRP replication service is advertised using DNS-SD [RFC6763]. The service name is '_srpl-tls._tcp'. Each SRP replication partner should have its own hostname, which when combined with the service instance name and the local DNS-SD domain name will produce a service instance name, for example 'example-host._srpl-tls._tcp.local.' The domain under which the service instance name appears will be 'local' for mDNS, and will be whatever domain is used for service registration in the case of a non-mDNS local DNS-SD service.

SRP replication uses DNS port 853 [RFC7858] and is based on DNS Stateful Operations [RFC8490]. Therefore, the SRV record for the example we've given would be:

```
example-host._srpl-tls._tcp.local. IN SRV 0 0 853 example-  
host.local.
```

The TXT record for SRP replication advertises the following fields:

dataset a 64-bit number encoded as hexadecimal ASCII, produced with a high-quality random number generator [RFC4086]. The dataset ID is used by SRP servers to establish a common SRP dataset for a domain.

join 'yes' or 'no'. Indicates whether other SRP replication servers are invited to join in replicating the dataset.

prec a 32-bit number representing the precedence of the server advertising the TXT record, represented in decimal

domain the domain name that this dataset is intended to represent

This identifier need not be persistent across SRP replication partner restarts. So in our example the TXT record might look like this:

```
#domain=openthread.thread.home.arpa.\025dataset=eb5bb51919a15cec\006p  
rec=1\008join=yes
```

(Note that each name/value pair in the TXT record is length-encoded, so the '#', the '\025', the '\006' and the '\008' are the lengths of the four name/value pairs.)

2.3. Discovering other replication services

SRP Replication is a cooperative process. In order to ensure cooperation between SRP replication partners on a link, it is necessary that each replication partner be aware of other potential partners. This is accomplished by maintaining a continuous browse for services of the service type "_srpl-tls._tcp".

An SRP Replication Partner MUST maintain an ongoing DNS-SD browse on the service name '_srpl-tls._tcp' within the local browsing domain. The ongoing browse will produce two different types of events: 'add' events and 'remove' events. When the browse is started, it should produce an 'add' event for every SRP replication partner currently present on the network, including the partner that is doing the browsing. Whenever a partner goes offline, a 'remove' event should be produced. 'remove' events are not guaranteed, however.

When a new service is added, the SRP partner checks to see if it is in a compatible domain. If the SRP partner has a domain to advertise, it compares that domain to the domain advertised in the added service instance: if they are not the same, then this instance is not a candidate for connection, and should be ignored.

2.3.1. Getting an SRP zone name from other SRP services

If the SRP partner does not have a domain to advertise, then when it begins to browse for partners, it sets a timer for DOMAIN_DISCOVERY_TIMEOUT seconds.

If the SRP partner does not have a domain to advertise, and is therefore willing to join an existing domain, it checks to see if the TXT record for the service indicates that joining is permitted. If so, the SRP partner adopts the provided domain name. Once it has adopted such a domain name, it updates its own TXT record to indicate that domain name, and sets the 'join=yes' key/value pair in the TXT record. It also cancels the DOMAIN_DISCOVERY_TIMEOUT timer.

If the `DOMAIN_DISCOVERY_TIMEOUT` timer goes off, then the SRP partner MUST propose a zone name using one of the methods mentioned previously in Section 2.1. It advertises that zone name in its TXT record, with `'join=yes'`. It then sets a new timer for `DOMAIN_PROPOSE_TIMEOUT` seconds.

While waiting for the `DOMAIN_PROPOSE_TIMEOUT` timer to go off, any new 'add' events that arrive are examined to see if they are potential domains to join. If a potential domain to join is seen, and it is the same as the proposed domain, then the partner adopts that domain and treats it as its domain to advertise. It then cancels the `DOMAIN_DISCOVERY_TIMEOUT` timer.

At this point the SRP replication partner has a domain to advertise: either the one it produced, or one that it discovered.

2.3.2. Establishing Replication with an existing Dataset

Once an SRP replication partner has settled on a domain to advertise, it must either join other SRP replication partners in replicating that domain, or if it is the first, it must advertise its willingness to participate in replicating the domain. In order to do this, it must settle on a dataset ID.

The dataset ID is a random 64-bit number, generated by the first server to offer that dataset. There should always be exactly one dataset ID per domain, but the dataset ID has a separate purpose: it represents the set of data that is being replicated by a set of cooperating SRP replication partners. This data is then offered under the agreed-upon domain, but it's possible that there might be several sets of SRP replication partners that agree to replicate a particular domain, and then some event occurs which renders these partners visible to each other. When this happens, the independent sets of partners must converge on a single dataset. This is done using the dataset ID.

When more than one dataset ID is present for a particular domain, the dataset ID that is numerically lowest is preferred. This means that SRP replication partners that are currently replicating a dataset with a numerically higher dataset ID will have to abandon that dataset and join together in replicating the numerically lowest dataset. Servers that are not replicating the numerically lowest dataset will therefore stop advertising SRP replication service and begin attempting to join in replicating the preferred dataset.

When a set of servers are advertising a particular dataset ID, the server with the lowest precedence is primary. The primary server is responsible for handing out precedence values to new partners as they

join in replicating the dataset. Precedence IDs are always allocated starting with the precedence that is one greater than the primary's precedence.

When an SRP replication partner has stopped advertising a particular dataset ID, or has just started and therefore hasn't started advertising a particular dataset ID, and there is a dataset ID present that it can join in replicating, it attempts to connect to the SRP replication partner that is primary for the dataset. If the startup handshake succeeds, the primary will assign a new precedence to the connecting partner as part of the handshake.

Once the synchronization phase has finished, the connecting partner will begin advertising the SRP service for the chosen domain using the new dataset ID and the precedence it received from the primary. The connecting server will then also attempt to connect to every SRP partner it sees advertising the same dataset ID and a lower precedence.

It is possible that an SRP partner will attempt to join in replicating a dataset, but the primary for that dataset may have discontinued service, but the advertisement for the primary is still in the cache. In this case, the SRP partner will attempt to reconfirm the primary's advertisement. In mDNS, this is done as described in Section 10.4 of [RFC6762]. For DNS Push connections, this is done using the RECONFIRM message, described in Section 6.5 of [RFC8765]. For regular (polled) DNS, the SRP partner must trigger a new DNS query. If the primary advertisement is successfully confirmed, this indicates that there is a problem connecting to the primary, in which case the connecting partner SHOULD discontinue attempting to connect for at least MIN_RECONNECT_AFTER_FAILURE seconds.

Otherwise, the connecting partner will attempt to connect to the new primary if there is one. If there are no other servers advertising the dataset ID, then the connecting partner reverts to attempting to start its own replication of that dataset.

2.3.3. Establishing Replication When There Is No Existing Dataset

When an SRP replication partner has attempted to discover partners with which to connect, and failed to do so, it then creates its own dataset ID and precedence and begins advertising that dataset. Both the dataset ID and precedence should be generated using a non-deterministic random number generator. The dataset ID should be a random number greater than or equal to zero and less than 2^{64} . The precedence should be a random number greater than or equal to 0 and less than 2^{15} . The reason for the upper limit is to allow for a large range of numbers toward which the precedence can increase.

The replication partner begins advertising this new dataset as soon as the dataset ID and precedence have been generated. As in the previous section, if a new dataset ID is seen shortly afterwards, this most likely indicates that two SRP replication instances came up at the same time; in this case as with the previous one, the lower dataset ID is preferred, and the partner advertising the higher dataset ID abandons that dataset ID to join the partner with the lower dataset ID.

The replication partner that first advertises the dataset is the primary replication partner for that dataset. It is responsible for assigning precedences to new partners.

2.3.4. Conflicting precedence values

It is possible that two SRP replication partners that see different service advertisements could identify different SRP replication servers as primary and attempt to get their precedence values from those different servers. When this happens, it's possible that they might both get the same precedence value. When this occurs, as soon each partner sees another partner advertising its precedence in an SRP replication advertisement, it must discontinue advertising and restart the dataset discovery process.

2.3.5. Handling primary transitions

An SRP partner either identifies itself as primary or not. When an SRP partner is primary, it never connects to other SRP servers--it only receives connections. When a non-primary partner connects to the primary partner, it knows it is connecting to the primary partner. If the connection with the primary drops, or if the primary's advertisement goes away, then the non-primary evaluates the set of advertisements that it sees. If its precedence is lowest, it identifies itself as primary.

Non-primary servers receive updates from the primary whenever the maximum precedence value changes. Non-primary servers should track this precedence value. When a non-primary becomes primary, it should add ten to the most recently received precedence value, so as to skip any possible precedence assignments that haven't yet propagated.

2.4. Discovering the addresses of partners

When a partner is discovered, two new ongoing mDNS queries are started on the hostname indicated in the SRV record of the partner: one for A records, and one for AAAA records. Each time an address 'add' event is seen, either for an 'A' record or an 'AAAA' record, the partner adds the address to the list of addresses belonging to that partner.

2.5. Establishing Communication with a replication partner

When an address is added to a partner's address list, the partner first checks to see if the address is one of its own addresses. If so, then the partner is marked "me", and no connection is attempted to it. This is somewhat safer than comparing hostnames, since a hostname collision can result in renaming.

If the partner is not marked 'me', then the partner checks to see if it has an existing outgoing connection to that partner. If it does not, then it checks to see whether it has disabled outgoing connections to that partner. If not, then it attempts to connect on the new address.

When a connection fails, it advances to the next address in the list, if there is one. If there are no remaining addresses, the partner sets a timer for RECONNECT_INTERVAL seconds. When this timer expires, it starts again at the beginning of the list and attempts to connect to the first address, iterating again across the list until a connection succeeds or it runs out of addresses.

Additionally, when an address is added, it is checked against the list of unidentified incoming connections. If a match is found, and the partner is marked "me," then the unidentified connection is removed from the list and dropped. Otherwise, it is attributed to the matching partner, and the protocol is started at the point of receiving an incoming connection.

When an outgoing connection succeeds, the partner sends its server ID.

2.6. Incoming connections

When an incoming connection is received, it is checked against the partner list based on the source address of the incoming connection. If the address appears on the list of addresses for a partner, then the connection is attributed to that partner. If no matching partner is found, a timer of UNIDENTIFIED_PARTNER_TIMEOUT seconds is set, and the incoming connection is added to the list of "unidentified" connections.

If a matching partner is found, then the partner waits for an incoming partner ID. When such an ID is received, it is compared to the partner's server-id. If the incoming server ID is the same as or greater than the partner's server ID, the connection is dropped. Otherwise, the connection proceeds to the "initial synchronization" state.

2.7. Eliminating extra connections

When an outgoing connection succeeds, the partner sends its server ID to the partner. When an incoming connection succeeds, the partner waits for a server ID. Because both connections are partner connections, and we only need one connection, the partner with the higher server ID acts as the client and the partner with the lower server ID acts as the server. If the server IDs are equal, then the connecting server generates a new server ID, updates its TXT record, and re-does the comparison.

2.8. Initial synchronization

The connecting partner begins the session by sending its server ID. The receiving partner waits for a server ID, and when it receives one, does the server ID comparison mentioned earlier. If the connection survives the comparison, then the server sends a response to the session message and waits for the client to request a list of update candidates.

The connecting partner waits for a response to the initial session message, and when it is received, requests that the server send candidates.

2.8.1. Sending candidates

When a partner receives a "send candidates" message that it is expecting to receive, it generates a candidate list from the list of known SRP clients. This list includes SRP clients that have registered directly with the partner, and SRP clients that have been received through SRP replication updates. Each candidate contains a hostname, a time offset, and a key identifier.

The key identifier is computed as follows:

```
uint32_t key_id(uint8_t *key_data, int key_len) {
    uint32_t key_id = 0;
    for (int i = 0; i < key_data_len; i += 4) {
        key_id += ((key_data[i] << 24) | (key_data[i + 1] << 16) |
                  (key_data[i + 2] << 8) | (key_data[i + 3]));
    }
    return key_id;
}
```

When a partner receives a candidate message during the synchronization process, it first searches for an SRP registration with a hostname that matches the hostname in the candidate message. It then compares the key ID to the key ID in the candidate message. If the key ID doesn't match, it sends back a candidate response status of "conflict". If the key ID does match, it compares the time provided to the time the existing host entry was received. If the time of the update is later, it sends a "send host" response. If it is earlier or the same, it sends a "continue" response. If there is no matching host entry for the candidate message, the partner sends a "send host" response.

When a partner receives a candidate response with a status of "send host", it generates a host message, which contains the hostname, the time offset, and the SRP message that was received from the host. The partner then applies the SRP update message as if it had been received directly from the SRP client. The host update time sent by the partner is remembered as the time when the update was received from the client, for the purposes of future synchronization.

When a partner is finished iterating across its list of candidates, it sends a "send candidates" response.

When a partner receives a "send candidates" response, if it is the server, it sends its own "send candidates" message, and processes any proposed candidates.

When a partner that is a server receives a "send candidates" response, it goes into the "routine operation" state. When a partner that is a client sends its "send candidates" response, it goes into the "routine operation" state.

2.9. Routine Operation

During routine operation, whenever an update is successfully processed from an SRP client, the partner that received that update queues that update to be sent to each partner to which it has a connection, whether server or client. If there are no updates pending to a particular client, the update is sent immediately. Otherwise, it's sent when the outstanding update is acknowledged.

When during routine operation a partner receives a host update from its partner, it immediately applies that update to its local SRP zone. This is based on the assumption that a new update is always more current than a copy of the host information in its database.

3. Protocol Details

The DNS-SD SRP Replication Protocol (henceforth SRPL) is based on DNS Stateful Operations [RFC8490]. Each SRP replication partner creates a listener on port 853, the DNS-over-TLS [RFC7858] reserved port. This listener can be used for other DNS requests as well.

Participants in the protocol are partners. To distinguish between partners, the terms "partner" and "partner" are used. "Partner" refers to the partner that is communicating or receiving communication. "Partner" refers to the other partner. Partners can be clients or servers: a partner that has established a connection to a partner is a client; a partner that has received a connection from a partner is a server.

3.1. DNS Stateful Operations considerations

DNS Stateful Operations is a DNS per-connection session management protocol. DNS Push session management includes session establishment as well as session maintenance.

3.1.1. DSO Session Establishment

An DSO session for an SRPL connection can be established either by simply sending the first SRPL message, or by sending a DSO Keepalive message. Section 5.1 of [RFC8490].

3.1.2. DSO Session maintenance

DSO sessions can be active or idle. As long as the SRPL protocol is active on a connection, the DSO state of the connection is active. DSO sessions require occasional keepalive messages. The default of fifteen seconds is adequate for SRPL.

An idle DSO session must persist for long enough that there is a chance for the browse that identifies it to succeed. Therefore, the minimum DSO session inactivity timeout is $2 * \text{UNIDENTIFIED_PARTNER_TIMEOUT}$ seconds.

3.2. DSO Primary TLVs

Each DSO message begins with a primary TLV, and contains secondary TLVs with additional information. The primary TLVs used in the SRPL protocol are as follows:

3.2.1. SRPL Session

DSO-TYPE code: `SRPLSession`. Introduces the SRPL session. The SRPL session TLV contains no data, just the type and length. SRPL Session primary TLV requests do not include any secondary TLVs. SRPL Session requests are DSO requests: the recipient is expected to send a response TLV. Both request and response TLVs have the same format. An SRPL Session response may include an SRPL Precedence secondary TLV.

3.2.1.1. SRPL client behavior

The SRPL Session request is sent by a partner acting as a client to its partner once the TLS connection to the partner, acting as a server, has succeeded. The SRPL session message establishes the DSO connection as an SRP protocol connection. If it is the first DSO message sent by the partner acting as a client, then it also establishes the DSO session.

When the SRPL partner acting as a client receives a response to its SRPL session message, it sends an SRPL Send Candidates message.

When an SRPL partner receives an SRPL Precedence secondary TLV in an SRPL Session response, if it thinks it is connected to the primary partner, it sets its precedence to the assigned value. If it thinks it is connecting to a non-primary, then it disconnects and waits `NON_PRIMARY_RESETTLE_TIME` seconds before reconnecting. It also attempts to reconfirm the service advertisement for the partner it thinks is primary.

3.2.1.2. SRPL server behavior

An SRPL partner acting as a server that receives an SRPL Session request checks to see if the connection on which it was received is already established. If so, this is a protocol error, and the SRPL partner MUST drop the connection.

When an SRPL partner that is primary receives an SRPL Session request, the SRPL Session response MUST include an SRPL Precedence secondary TLV, which assigns a precedence to the connecting SRPL partner.

3.2.2. SRPL Send Candidates

DSO-TYPE code: SRPLSendCandidates. Requests the partner to send its candidates list. The SRPL Send Candidates message contains no additional data. The SRPL Send Candidates primary TLV does not include any secondary TLVs. SRPL Send Candidates messages are DSO requests: the recipient is expected to send a response TLV. Both request and response TLVs have the same format.

3.2.2.1. SRPL client behavior

An SRPL partner acting as a client MUST send an SRPL Send Candidates request after it has received an SRPL Session response. It MUST NOT send this request at any other time.

An SRPL partner acting as a client expects to receive an SRPL Send Candidates message after it has received an SRPL Send Candidates response. If it receives an SRPL Send Candidates message at any other time, this is a protocol error, and the SRPL partner should drop its connection to the server.

3.2.2.2. SRPL server behavior

An SRPL partner acting as a server expects to receive an SRPL Send Candidates request after it has sent an SRPL Session response. If it receives an SRPL Candidates request at any other time, this is a protocol error, and it MUST drop the connection.

An SRPL partner acting as a server MUST send an SRPL Send Candidates request after it has sent an SRPL Send Candidates response.

An SRPL partner acting as a server MUST enter the "normal operations" state after receiving an SRPL Send Candidates response from its partner.

3.2.3. SRPL Candidate

DSO-TYPE code: SRPLCandidate. Announces the availability of a specific candidate SRP client registration. The SRPL Candidate message contains no additional data. SRPL Candidate messages are DSO requests: the recipient is expected to send a response TLV. Both request and response TLVs have the same format.

3.2.3.1. Required secondary TLVs

The SRPL Candidate request MUST include the following secondary TLVs: SRPL Hostname, SRPL Time Offset, and SRPL Key ID. If an SRPL partner receives an SRPL Candidate request that doesn't contain all of these secondary TLVs, this is a protocol error, and the partner MUST drop the connection.

The SRPL Candidate response MUST include one of the following status TLVs: SRPL Candidate Yes, SRPL Candidate No, or SRPL Conflict. If an SRPL partner receives an SRPL Candidate response which does not contain exactly one of these TLVs, this is a protocol error, and the partner MUST drop the connection.

3.2.3.2. SRPL partner common behavior

SRPL partners expect to receive SRPL Candidate messages between the time that they have sent an SRPL Send Candidates message and the time that they have received an SRPL Send Candidates response. If an SRPL Candidate message is received at any other time, this is a protocol error, and the partner MUST drop the connection.

Partners MUST NOT send SRPL Candidate requests if they have sent any SRPL Candidate or SRPL host requests that have not yet received responses. Partners receiving SRPL Candidate requests when they have not yet responded to an outstanding SRPL Candidate request or SRPL Host request MUST treat this as a protocol failure and drop the connection.

When a partner receives a valid SRPL Candidate message, it checks its SRP registration database for a host that matches both the SRPL Hostname and SRPL Key ID TLVs. If such a match is not found, the partner sends an SRPL Candidate response that includes the SRPL Candidate Yes secondary TLV.

If a match is found for the hostname, but the Key ID doesn't match, this is a conflict, and the partner sends an SRPL Candidate response with the SRPL Conflict secondary TLV.

If a match is found for the hostname, and the key ID matches, then the partner computes the update time of the candidate by subtracting the value of the SRPL Time Offset TLV from the current time in seconds. This computation should be done when the SRPL Candidate message is received to avoid clock skew. If 'candidate update time' - 'local update time' is greater than SRPL_UPDATE_SKEW_WINDOW, then the candidate update is more recent than the current SRP registration. In this case, the partner sends an SRPL Candidate response and includes the SRPL Candidate Yes secondary TLV. The reason for adding in some skew is to account for network transmission delays.

3.2.4. SRPL Host

DSO-TYPE code: SRPLHost. Provides the content of a particular SRP client registration. The SRPL Host message contains no additional data. SRPL Host messages are DSO requests: the recipient is expected to send a response TLV. Both request and response TLVs have the same format.

3.2.4.1. Required secondary TLVs

The SRPL Host request MUST include the following secondary TLVs: SRPL Hostname, SRPL Key ID, and one or more SRPL Host Message TLVs. If an SRPL partner receives an SRPL Candidate request that doesn't contain all of these secondary TLVs, this is a protocol error, and the partner MUST drop the connection.

The SRPL Host message always includes at least one SRPL Host Message TLV, which contains the most recent update the SRP server has received for that host. However, in some cases an update for a host may update some, but not all, service instances that reference that host; in this case, the SRPL Host request MUST include all of the previously received SRP updates that would be required to reconstruct the current state of the host registration on the server sending the SRPL Host request.

3.2.4.2. SRPL partner common behavior during synchronization

SRPL partners expect to receive either zero or one SRPL Host requests after sending an SRPL Candidate response with a SRPL Candidate Yes secondary TLV. If an SRPL Host request is received at any other time during synchronization, this is a protocol error, and the partner MUST drop the connection. The only time that an SRPL Host request would not follow a positive SRPL Candidate response would be when the candidate host entry's lease expired after the SRPL Candidate request was sent but before the SRPL Candidate response was received.

SRPL partners send SRPL Host requests during synchronization when a valid SRPL Candidate response has been received that includes an SRPL Candidate Yes secondary TLV. The host request is generated based on the current candidate (the one for which the SRPL Candidate request being responded to was sent).

3.2.4.3. SRPL partner common behavior during normal operations

When an SRPL partner during normal operations receives and has successfully validated an SRP update from an SRP client, it MUST send that update to each of its connected partners as an SRPL Host request. If the connection to a particular partner is not busy, and there are no updates already queued to be sent, it MUST send the SRPL Host message immediately. Otherwise, it MUST queue the update to send when possible. The queue MUST be first-in, first-out.

After an SRPL partner has sent an SRPL Host request to a partner, and before it receives a corresponding SRPL Host response, it MUST NOT send any more SRPL Host messages to that partner.

When an SRPL partner receives an SRPL Host request during normal operations, it MUST apply it immediately. While it is being applied, it MUST NOT send any other SRPL Host requests to that partner.

When an SRPL Host request has been successfully applied by an SRPL partner, the partner MUST send an SRPL Host response.

If an SRPL partner receives an SRPL Host request while another SRPL Host request is being processed, this is a protocol error, and the partner MUST drop the connection to its partner.

3.3. DSO Secondary TLVs

In addition to the Primary TLVs used to send requests between SRPL partners, we define secondary TLVs to carry formatter information needed for various SRPL requests.

3.3.1. SRPL Candidate Yes

DSO-TYPE code: SRPLCandidateYes. In an SRPL Candidate response, indicates to the partner that an SRPL Host message for the candidate is wanted and should be sent.

Appears as a secondary TLV in SRPL Candidate responses. MUST NOT appear in any other SRPL request or response. MUST NOT appear in addition to either SRPL Conflict or SRPL Candidate No secondary TLVs.

3.3.2. SRPL Candidate No

DSO-TYPE code: SRPLCandidateNo. In an SRPL Candidate response, indicates to the partner that an SRPL Host message for the candidate is not wanted and should not be sent.

Appears as a secondary TLV in SRPL Candidate responses. MUST NOT appear in any other SRPL request or response. MUST NOT appear in addition to either SRPL Conflict or SRPL Candidate Yes secondary TLVs.

3.3.3. SRPL Conflict

DSO-TYPE code: SRPLConflict. In an SRPL Candidate response, indicates to the partner that an SRPL Host message for the candidate is not wanted and should not be sent. Additionally indicates that the proposed host conflicts with local data. This indication is informative and has no effect on processing.

Appears as a secondary TLV in SRPL Candidate responses. MUST NOT appear in any other SRPL request or response. MUST NOT appear in addition to either SRPL Candidate Yes or SRPL Candidate No secondary TLVs.

3.3.4. SRPL Hostname

DSO-TYPE code: SRPLHostname. In an SRPL Candidate or SRPL Host request, indicates to the partner the hostname of an SRP registration. The hostname is represented in DNS wire format Section 3.1 of [RFC1035].

Required as a secondary TLV in SRPL Candidate and SRPL Host requests. MUST NOT appear in any other SRPL request or response.

3.3.5. SRPL Host Message

DSO-TYPE code: SRPLHostMessage. In an SRPL Host request, conveys four data objects in order:

- * the lease time and key lease time returned to the client, represented as two unsigned 32-bit numbers in units of seconds.
- * the time offset at which the message was received, represented as a 32-bit unsigned number of seconds. The time offset is computed as the difference between the time when the SRPL Host Message TLV is being constructed for transmission, and the time when the SRP update contained in the SRPL Host Message was received.

- * the SRP Update message received from the SRP client. This contains the contents of the message, but not any IP, UDP, TCP or TLS headers that may have encapsulated it.

The content of the SRPL Host Message is used to update the host on the partner receiving the request. Note that the SRP message being sent can't be modified by the SRPL partner sending it, so in order to validate the message (assuming that the signature includes a nonzero time), the validation process should adjust the current time by the time offset included in the SRPL Time Offset TLV when comparing against the signature time when checking for replay attacks. The computation of the current time of signing should be done when the message is received to avoid clock skew that might result from processing delays.

Required as a secondary TLV in SRPL Host requests. MUST NOT appear in any other SRPL request or response.

3.3.6. SRPL Time Offset

DSO-TYPE code: SRPLTimeOffset. In an SRPL Candidate, conveys the difference between the time the SRP update was received from the SRP client and the current time on the partner generating the request, in seconds.

Required as a secondary TLV in SRPL Candidate and SRPL Host requests. MUST NOT appear in any other SRPL request or response.

3.3.7. SRPL Key ID

DSO-TYPE code: SRPLKeyID. In an SRPL Candidate, conveys the key ID of the SRP client.

Required as a secondary TLV in SRPL Candidate requests. MUST NOT appear in any other SRPL request or response.

4. Security Considerations

SRP replication basically relies on the trustworthiness of hosts on the local network. Since SRP itself relies on the same level of trust, SRP replication doesn't make things worse. However, when the option to have a central SRP server is available, that is likely to be more trustworthy.

5. Delegation of 'local.arpa.'

In order to be fully functional, the owner of the 'arpa.' zone must add a delegation of 'local.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].

6. IANA Considerations

6.1. 'srpl-tls' Service Name

IANA is requested to add a new entry to the Service Names and Port Numbers registry for srpl-tls 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-SD SRP Replication Service for a given domain is advertised using the "_srpl-tls._tcp.<domain>." SRV record gives the target host and port where DNS-SD SRP Replication Service is provided for the named domain.

6.2. DSO TLV type code

The IANA is requested to add the following entries to the 16-bit DSO Type Code Registry. Each type mnemonic should be replaced with an allocated type code, both in this table and elsewhere in the document. RFC-TBD should be replaced with the name of this document once it becomes an RFC.

Type	Name	Early Data	Status	Reference
SRPLSession	SRPL Session	No	STD	RFC-TBD
SRPLSendCandidates	SRPL Send Candidates	No	STD	RFC-TBD
SRPLCandidate	SRPL Candidate	No	STD	RFC-TBD
SRPLHost	SRPL Host	No	STD	RFC-TBD
SRPLCandidateYes	SRPL Candidate Yes	No	STD	RFC-TBD
SRPLCandidateNo	SRPL Candidate No	No	STD	RFC-TBD
SRPLConflict	SRPL Conflict	No	STD	RFC-TBD
SRPLHostname	SRPL Hostname	No	STD	RFC-TBD
SRPLHostMessage	SRPL Host Message	No	STD	RFC-TBD
SRPLTimeOffset	SRPL Time Offset	No	STD	RFC-TBD
SRPLKeyID	SRPL Key ID	No	STD	RFC-TBD

Table 1

6.3. Registration and Delegation of 'local.arpa' as a Special-Use Domain Name

IANA is requested to record the domain name local.arpa.' in the Special-Use Domain Names registry [SUDN]. IANA is requested, with the approval of IAB, to implement the delegation requested in Section 5.

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 local.arpa.' with the description "Ad-hoc DNS-SD Special-Use Domain", listing this document as the reference.

7. Informative References

8. 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>>.
- [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>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.

Author's Address

Ted Lemon
Apple Inc.
One Apple Park Way

Cupertino, California 95014
United States of America

Email: mellon@fugue.com

DNSSD
Internet-Draft
Intended status: Standards Track
Expires: 13 January 2022

S. Cheshire
T. Lemon
Apple Inc.
12 July 2021

Advertising Proxy for DNS-SD Service Registration Protocol
draft-sctl-advertising-proxy-02

Abstract

An Advertising Proxy allows a device that accepts service registrations using Service Registration Protocol (SRP) to make those registrations visible to legacy clients that only implement Multicast DNS.

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 13 January 2022.

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	2
1.1. Conventions and Terminology Used in This Document	3
2. Advertising Proxy	3
2.1. Name Conflicts	3
2.1.1. Name Conflicts in Managed Namespaces	5
2.2. Data Translation	6
2.3. No Text-Encoding Translation	6
2.4. No Address Suppression	6
2.5. No Support for Reconfirm	7
3. Security Considerations	8
4. IANA Considerations	8
5. References	8
5.1. Normative References	8
5.2. Informative References	9
Authors' Addresses	10

1. Introduction

DNS-Based Service Discovery [RFC6763] [ROADMAP] was designed to facilitate Zero Configuration IP Networking [RFC6760] [ZC]. When used with Multicast DNS [RFC6762] with ".local" domain names [RFC6761] this works well on a single link (a single broadcast domain).

There is also a desire to have DNS-Based Service Discovery work between multiple links that aren't part of the same broadcast domain [RFC7558]. Even within a single Wi-Fi broadcast domain it is beneficial to reduce multicast traffic, because, in comparison to Wi-Fi unicast traffic, Wi-Fi multicast is inefficient, slow, and unreliable [MCAST].

There are three complementary ways that this move towards decreased reliance on multicast is achieved.

One variant is pure end-to-end unicast, with services using unicast Service Registration Protocol [SRP] to register with a service registry, and clients using unicast DNS Push Notification subscriptions [RFC8765] over DNS Stateful Operations [RFC8490] to communicate with the service registry to discover and track changes to those registered services.

A second variant is a hybrid approach that facilitates legacy devices that only implement link-local Multicast DNS (like your ten-year-old network laser printer) having their services discovered by remote clients using a unicast DNS Push Notifications session to a Discovery Proxy [RFC8766].

The third variant, documented here, is a logical complement to the second variant. It enables legacy clients (that only implement link-local Multicast DNS) to discover services registered (using unicast) with a service registry. The service registry accepts service registrations using unicast Service Registration Protocol [SRP], and makes those service registrations visible, both to remote clients using unicast DNS Push Notifications [RFC8765] and, using the Advertising Proxy mechanism documented here, to local clients using Multicast DNS [RFC6762].

1.1. Conventions and Terminology Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Advertising Proxy

An Advertising Proxy can be a component of any DNS authoritative server, though it logically makes most sense as a component of a service registry (a DNS authoritative server that implements Service Registration Protocol [SRP]). A client can send registration requests for any valid DNS records to a service registry, though in practice the most common use is to register the PTR, SRV and TXT records that describe a DNS-SD service [RFC6763], and the A and AAAA records that give the IPv4 and IPv6 addresses of the target host where that service can be reached.

When a service registry accepts a registration request for DNS records, a service registry that implements an Advertising Proxy also advertises equivalent records using Multicast DNS on one or more configured local multicast-capable interfaces. An Advertising Proxy could also advertise on one or more configured remote multicast-capable interfaces using a Multicast DNS Relay [RELAY]. For the purposes of this document, a local multicast-capable interface directly attached to the host and a remote multicast-capable interface connected via a relay are considered to be equivalent.

2.1. Name Conflicts

In the event that an SRP client attempts to register a record with a name that was already created in that registry by a different SRP client, or is otherwise disallowed by policy, a name conflict is reported and the new client is required to choose a new name.

Similarly, Multicast DNS implements first-come-first-served name allocation. When a registered record is advertised using Multicast DNS it may suffer a name conflict if a conflicting Multicast DNS record with that name already exists on that link. In the case of network failure and subsequent recovery, Multicast DNS can also signal name conflicts at a later time during the life of a record registration. For example, if the network link is partitioned at the time of record registration, the name conflict may not be discovered until later when the partition is healed.

Specifically, a name conflict can occur:

1. During the SRP validation process, because another SRP client has already registered the same name.
2. Immediately while the Advertising Proxy is registering the name, if the Multicast DNS uniqueness probes detect a conflicting record.
3. After the name has been successfully registered, but before the response has been sent to the client.
4. After the initial response has been sent to the client.

In the first three cases, the client can be notified of the conflict at the time of registration, and is expected to choose a new name. In the last case, SRP clients must be coded defensively to handle the case where an apparently successful record registration is later determined to be in conflict, just as existing Multicast DNS clients have to be coded defensively to handle late conflicts gracefully. With a sleepy SRP client there may be no way to notify it of the conflict until it next re-registers. In the case of late conflicts, the service registry with Advertising Proxy capability is responsible for selecting a temporary new name to be used until the client renews. When the client next renews, the service registry informs the client of the new name the service registry selected on its behalf. The client can choose to accept that new name, or select a new name of its own choosing.

The registration process has several steps. First the hostname claimed by the SRP client must be registered. Once this has succeeded, the Advertising Proxy can register all of the service instances that point to that hostname. When all of these registrations have succeeded, the service registry can finally send its response to the SRP client. If any of them fail, they must all be removed and the client notified of the failure. If the failure is a result of a name conflict, the response code should be YXDOMAIN. Other SRP failures are documented in the SRP specification. Any other failures not contemplated in the SRP specification return SERVFAIL.

2.1.1. Name Conflicts in Managed Namespaces

In some cases, the name conflict resolution behavior described above is neither needed nor desirable. For instance, when the set of expected SRP clients is known to include only clients added with some kind of commissioning or on-boarding protocol that guarantees that hostnames are unique, it may cause serious problems to rename such a device.

In this situation, the Advertising Proxy behavior should be different: it should be assumed that all names registered with SRP that survive SRP's first-come, first-serve name conflict detection are indeed as intended. Any conflict that may be discovered as a result of advertising those names using mDNS can be assumed to either be an error or an attack, and there is no benefit to renaming such a device: it will not be usable under its new name.

In this case, the Advertising Proxy simply performs normal SRP first-come, first-serve naming and then updates its local idea of the SRP namespace. This update is then reflected in mDNS. If a conflict is detected, the Advertising Proxy schedules a new attempt to claim the name at some time in the future: long enough that these re-attempts to not generate excessive multicast traffic, but short enough that an accidental conflict is cured in a reasonable timeframe.

The downside to this approach is that if the device on the multicast network persists in claiming the name, the SRP client that claimed it will be unreachable. Networks that use Advertising Proxies configured to behave in this way should provide a way to rename the device that is suffering the conflict. However, if the failure is the result of a malicious attack by a device on the multicast network, that device will have to be identified and removed before the attack can be eliminated.

In order to address this problem, it may be advisable to provide with a way for the advertising proxy to inform the mDNS service that it should continue to advertise the name that is in conflict, rather than ceasing to do so when the conflict is detected.

2.2. Data Translation

As with a Discovery Proxy [RFC8766] some translation needs to be performed before the Advertising Proxy makes the registered unicast data visible using Multicast DNS. Specifically, the unicast DNS domain name suffix configured for Advertising Proxy use is stripped off and replaced with the top-level label "local".

2.3. No Text-Encoding Translation

As with a Discovery Proxy [RFC8766], an Advertising Proxy does no translation between text encodings [RFC6055]. Specifically, an Advertising Proxy does no translation between Punycode encoding [RFC3492] and UTF-8 encoding [RFC3629], either in the owner name of DNS records or anywhere in the RDATA of DNS records (such as the RDATA of PTR records, SRV records, NS records, or other record types like TXT, where it is ambiguous whether the RDATA may contain DNS names). All bytes are treated as-is with no attempt at text-encoding translation. A server implementing DNS-based Service Discovery [RFC6763] will use UTF-8 encoding for its unicast DNS-based record registrations, which the Advertising Proxy passes through without any text-encoding translation to the Multicast DNS subsystem. Queries from peers on the configured multicast-capable interface are answered directly from the advertised data without any text-encoding translation.

2.4. No Address Suppression

Unlike a Discovery Proxy [RFC8766], an Advertising Proxy does not need to selectively suppress link-local [RFC3927] [RFC4862] or other addresses. Since the clients of the service registry are registering their records in a unicast DNS namespace, there is a presumption they they will only register addresses with sufficient scope to be usable by the anticipated clients. No further filtering or suppression by the service registry is required. In most cases it is acceptable for devices registering with a service registry to register all of their available addresses, and a client implementing Happy Eyeballs [RFC8305] connecting to that service will automatically select an appropriate address to use.

2.5. No Support for Reconfirm

For network efficiency, Multicast DNS [RFC6762] uses fairly long record lifetimes (typically 75 minutes). When a client is unable to reach a service that it discovered, Multicast DNS provides a "reconfirm" mechanism that enables the client to signal to the Multicast DNS subsystem that its cached data may be suspect, which causes the Multicast DNS subsystem to reissue queries, and remove the stale records if the queries are not answered.

Similarly, when using unicast service discovery with a Discovery Proxy [RFC8766], the DNS Push Notifications [RFC8765] protocol provides the RECONFIRM mechanism to signal that the Discovery Proxy should perform a local Multicast DNS reconfirm operation to re-verify the validity of the records.

When an Advertising Proxy is used, to support legacy clients that only implement Multicast DNS, reconfirm operations have no effect. If a device uses unicast Service Registration Protocol [SRP] to register its services with a service registry with Advertising Proxy capability, and the device then gets disconnected from the network, the Advertising Proxy will continue to advertise those records until the registrations expire. If a client discovers the service instance using Multicast DNS and is unable to reach it, and uses a Multicast DNS reconfirm operation to re-verify the validity of the records, then the Advertising Proxy will continue to answer on behalf of the departed device until the record registrations expire. The Advertising Proxy has no reliable way to determine whether the additional Multicast DNS queries are due to a reconfirm operation, or due to other routine causes, like a client being rebooted, or disconnecting and then reconnecting to the network. The service registry has no reliable automatic way to determine whether a device that registered records has failed or disconnected from the network. Particularly with sleepy battery powered devices, the service registry does not know what active duty cycle any given service is expected to provide.

Consequently, reconfirm operations are not supported with an Advertising Proxy. In cases where use of the reconfirm mechanism is important, clients should be upgraded to use the unicast DNS Push Notifications [RFC8765] protocol's RECONFIRM message. This RECONFIRM message provides an unambiguous signal to the service registry that it may be retaining stale records. (A future update to the Service Registration Protocol document [SRP] will consider ways that this unambiguous signal can be used to trigger expedited removal of stale data.)

3. Security Considerations

An Advertising Proxy may made data visible to eavesdroppers on the configured multicast-capable link(s).

4. IANA Considerations

This document has no IANA actions.

5. References

5.1. Normative References

- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC8765] Pusateri, T. and S. Cheshire, "DNS Push Notifications", RFC 8765, DOI 10.17487/RFC8765, June 2020, <<https://www.rfc-editor.org/info/rfc8765>>.

- [SRP] Lemon, T. and S. Cheshire, "Service Registration Protocol for DNS-Based Service Discovery", Work in Progress, Internet-Draft, draft-ietf-dnssd-srp-09, 11 January 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-dnssd-srp-09>>.

5.2. Informative References

- [RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, DOI 10.17487/RFC3492, March 2003, <<https://www.rfc-editor.org/info/rfc3492>>.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <<https://www.rfc-editor.org/info/rfc3629>>.
- [RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, DOI 10.17487/RFC3927, May 2005, <<https://www.rfc-editor.org/info/rfc3927>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC6055] Thaler, D., Klensin, J., and S. Cheshire, "IAB Thoughts on Encodings for Internationalized Domain Names", RFC 6055, DOI 10.17487/RFC6055, February 2011, <<https://www.rfc-editor.org/info/rfc6055>>.
- [RFC7558] Lynn, K., Cheshire, S., Blanchet, M., and D. Migault, "Requirements for Scalable DNS-Based Service Discovery (DNS-SD) / Multicast DNS (mDNS) Extensions", RFC 7558, DOI 10.17487/RFC7558, July 2015, <<https://www.rfc-editor.org/info/rfc7558>>.
- [RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <<https://www.rfc-editor.org/info/rfc8305>>.
- [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>>.

- [MCAST] Perkins, C. E., McBride, M., Stanley, D., Kumari, W., and J. C. Zuniga, "Multicast Considerations over IEEE 802 Wireless Media", Work in Progress, Internet-Draft, draft-ietf-mboned-ieee802-mcast-problems-13, 4 February 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-mboned-ieee802-mcast-problems-13>>.
- [RELAY] Lemon, T. and S. Cheshire, "Multicast DNS Discovery Relay", Work in Progress, Internet-Draft, draft-ietf-dnssd-mdns-relay-04, 22 February 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-dnssd-mdns-relay-04>>.
- [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>>.
- [ZC] Cheshire, S. and D. H. Steinberg, "Zero Configuration Networking: The Definitive Guide", O'Reilly Media, Inc., ISBN 0-596-10100-7, December 2005.

Authors' Addresses

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

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

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

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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 27 January 2022

T. Lemon
(J. Deng)
Apple Inc.
26 July 2021

Permissionless Advertising and Discovery of DNS-SD Authoritative Zones
draft-tlj-dnssd-zone-discover-00

Abstract

This document describes how to make DNS-SD browsing domains available for browsing and discovery without requiring special cooperation from the network infrastructure. Zones made available in this way are browsed using DNS or DNS Push. The mechanism for advertising them is Multicast DNS (mDNS). This allows DNS-SD browsers to benefit from the permissionless aspects of mDNS without relying on mDNS for all queries, which improves scalability and reliability in applications where many services may be advertised.

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 27 January 2022.

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	2
2. Glossary	3
3. Data Model	4
3.1. Authority Datasets	4
3.1.1. Multicast DNS	4
3.1.2. Authoritative DNS	4
3.1.3. SRP Replication	5
3.2. DNSSD Browsing Domains	5
4. Advertising an Authority Dataset to be Used for Service Discovery	5
4.1. Choosing a Domain to Advertise	6
4.2. Advertising DNS Authoritative Service for a Domain using multicast DNS	7
4.3. Advertising Address information for an Authority	8
4.4. Advertising the Availability of Service Discovery for a Domain	8
5. Discovering Authority Datasets	8
6. Security Considerations	10
7. Informative References	10
Authors' Addresses	10

1. Introduction

DNS-SD currently provides permissionless advertising and discovery using multicast DNS [RFC6762]. Unfortunately, multicast DNS has some limitations. In addition to the obvious limitation that it only works between services and users that are connected to a single multicast domain (generally a single link), in many situations excessive use of multicast is unreliable, which can cause discovery to fail when a service is actually present.

By contrast, DNS service discovery using unicast traffic is not limited by the scope of reachability of link-local multicast. On networks where multicast is less reliable, or more costly, unicast DNS-SD is clearly advantageous. However, because of the hierarchical nature of DNS, existing solutions for providing unicast DNS rely on coordination with the network infrastructure. In many settings, particularly on home networks, this coordination is not available, and so we must fall back on multicast DNS, despite its limitations.

This document defines a new mechanism for discovering the availability of unicast DNS service discovery using multicast DNS. Although somewhat limited in the sense that this mechanism still relies on multicast DNS as a means of discovering the unicast service, this mechanism can substantially reduce reliance on multicast DNS, so that its limitations are minimized. Additionally, it makes it possible for devices that provide discovery to adjacent networks, such as stub networks, to overcome the limitations of link-local multicast for this application.

This document describes how to advertise and discover authoritative service for a DNS domain, and how to advertise the availability of service discovery for a DNS domain, using multicast DNS.

2. Glossary

Advertiser a service that is advertising its availability through some authority

Authority a device, connected to an infrastructure link, that is advertising service discovery for an authority dataset using mDNS.

Authority Dataset a collection of authoritative data to be advertised, and which can be treated as a single coherent set. More than one authority may advertise the same dataset; what makes it "the same dataset" is the expectation that whichever authority is asked for an answer to a particular question will generally give the same answer as any other authority.

DNSSD Browser a device, connected to an infrastructure link, that discovers authorities and uses them to discover services advertised by advertisers.

3. Data Model

The goal of the mechanism described in this document is to enable permissionless advertising and discovery of authoritative DNS servers that can be used for service discovery. From the perspective of a consumer of such a service on a particular IP link, there may be more than one such service. Each such service can be thought of as providing an authority dataset.

3.1. Authority Datasets

3.1.1. Multicast DNS

One example of an authority dataset is the set of services that can be discovered by a DNSSD Discovery Proxy [RFCxxx]. A discovery proxy acts as an authoritative DNS server mapping information advertised using multicast DNS on a particular link to a DNS zone.

There may be more than one discovery proxy for a particular multicast DNS link. If so, both discovery proxies can be expected to return the same answers to any questions asked by a DNS-SD client that is browsing for services within that zone. This is what is meant by an "authority dataset": the data returned by one authoritative server that answers for that dataset should be the same as the data returned by the other server.

Because of the dynamic nature of mDNS, there is no enforcement mechanism to ensure that two discovery proxies would answer the same DNS question in exactly the same way, but each server is functionally equivalent: there is no reason to prefer one server over the other, nor to query both servers to avoid missing data known to one but not the other.

3.1.2. Authoritative DNS

Another example of an authority dataset is an authoritative DNS server that maintains a DNS zone for DNS Service Discovery using the DNS-SD Service Registration Protocol. In this case, there is one primary authoritative server and, potentially, more than one secondary authoritative server. The secondary server databases are all dependent on the primary server's database, and are maintained using DNS zone transfers or some other hierarchical replication mechanism.

In this case, the primary and the secondary servers are all serving an authoritative zone, which is another example of an authority dataset. The authoritative zone may have different versions, which can be known using the zone serial number, but in principle each

authoritative server is equivalently valid: there is no reason to prefer one over the other. This is another example of an authority dataset.

3.1.3. SRP Replication

A third example of an authority dataset would be a set of one or more SRP servers that cooperate to maintain a common database using the SRP replication protocol [SRP Replication]. These servers are each authoritative DNS servers, in the sense that they answer authoritatively for questions within the DNS zone that they manage, but unlike a typical DNS authoritative service configuration, there is no hierarchy—no server is primary—and there are no discrete versions of the zone database, so there is no way to generate a meaningful serial number that could be used to manage zone transfers.

As with Discovery Proxies, although it's quite possible at any given moment in time that the same query to two different SRP replication peers will yield different answers, the dataset being managed by these servers is the same dataset, and therefore is also an authority dataset.

3.2. DNSSD Browsing Domains

Service discovery on a local link always implicitly includes one authority dataset: the set of all mDNS services advertised on that link. DNSSD browsers always search for services within this dataset. Additional datasets are made available by advertising additional legacy browsing domains locally. So each authority dataset other than the link-local authority dataset must be explicitly advertised using mDNS; when the DNSSD browser is asked to browse for local services without explicitly specifying a domain in which to browse, it attempts to discover that service in each of the authority datasets advertised locally for discovery, and also in the link-local dataset provided by mDNS.

Note that DNSSD [RFC6763] also provides for discovering browsing domains using DNS, either using the domain name search list or using the DNS reverse domain query [RFC6763 section ???]. DNS browsing domains are provided by the network infrastructure, and complement browsing domains that may be provided permissionlessly using mDNS.

4. Advertising an Authority Dataset to be Used for Service Discovery

There are four steps an authority must follow to advertise the availability of an authority dataset for service discovery:

- * Choose a domain to represent that authority dataset

- * Advertise itself as an authority that provides name service for that domain (and hence that dataset)
- * Advertise its address information
- * Advertise the availability of service discovery for that domain

4.1. Choosing a Domain to Advertise

When advertising a domain for discovery, it must be the case that all authorities servers that advertise that domain are advertising the same information. Thus, the domain being advertised can be treated as an identifier for a particular authority dataset. How this is accomplished is out of scope for this document; one solution is described in [SRP Replication].

Because the domain is being advertised using multicast DNS, we assume that there is no delegation in the global DNS; if there were, there would be no reason to advertise the domain using mDNS. Furthermore, in order to prevent domain spoofing using the technique described here, the DNS resolver that is discovering this domain is required to prove that no delegation for the domain being advertised using mDNS exists in the global DNS hierarchy.

Given that most stub resolvers at present do not support DNSSEC, the domain being advertised will have to be a subdomain of some domain that is known to be a locally-served domain [RFC6761?]. In this case the client can be sure that this domain never appears in the DNS by definition, rather than by validating the non-existence of the delegation.

Two domains that are ideal for this purpose are 'home.arpa' [Dot Home] and 'service.arpa' [SRP]. The 'home.arpa' domain is generally intended for use in home networks, so this is appropriate for use in cases where the device advertising the domain is expected to be installed in a home network; for devices that are not expected to be installed in a home network, 'service.arpa' is preferable.

Given that there is no way for a particular device to know for certain that the network setting in which it is installed is in fact 'a home' or 'not a home,' this advice is merely a suggestion: a consumer product should probably use 'home.arpa' and a commercial product should probably use 'service.arpa', and perhaps either device should be configurable, but in practice it is not crucial to get this perfectly correct.

Bearing in mind that there may be multiple authorities and multiple authority datasets being provided on the same infrastructure link, it is important to choose a domain that is not ambiguous. Therefore, devices advertising domains for discovery MUST NOT use 'home.arpa' or 'service.arpa' directly: when using these domains, a unique subdomain must be chosen below that domain, rather than using the root domain.

It may be useful to identify the subdomain in terms of some visible network identifier. For instance, if the authority dataset contains the set of services for a particular WiFi link, it might make sense to use the name 'SSID.wifi.home.arpa'. This shows the SSID of the WiFi link, and differentiates between WiFi and other technologies (e.g. Thread) that use something like an SSID. For Thread networks, the domain 'thread.home.arpa' is used for this purpose, for example, and the thread network name is used as the leftmost label (e.g., 'my-home.thread.home.arpa').

We have stated that the domain must be provably nonexistent, which is a slight simplification. For completeness, we will point out that if the delegation is secure, and the server being advertised is able to sign records such that they validate, this is also permissible. But in this case, there's likely no utility to using mDNS to advertise the authoritative server and, furthermore, this solution requires the stub resolver to do DNSSEC validation, which is not commonly supported at present.

Although '.local' is a locally-served domain, it is by definition served using multicast DNS. For this reason, authorities MUST NOT use '.local' to advertise their authority dataset.

4.2. Advertising DNS Authoritative Service for a Domain using multicast DNS

To advertise DNS authoritative service for a domain, the authoritative server publishes an NS record for that domain using multicast DNS. For example, to publish DNS service for example.thread.home.arpa., the NS record published with mDNS would be:

```
example.thread.home.arpa. IN NS thread-server-1.local
```

Note that the DNS server for example.com has a .local suffix, meaning that its address can be discovered using mDNS. This is not required. If the DNS server is in a domain that can be looked up using ordinary DNS service, multicast DNS service is not required. This solution is preferred, but requires coordination with infrastructure, so doesn't address the core use case of this document.

Before publishing its chosen domain, the authority MUST validate that no other authority is advertising that domain for a different authority dataset. The mechanism for this validation is out of scope for this document, and is specific to the replication mechanism being used. If no replication mechanism is being used, the authority MUST publish its NS record as a unique record.

4.3. Advertising Address information for an Authority

Address information for an authority may be advertised using DNS or mDNS. If the authority happens to have a name published in the DNS, it SHOULD use that name, since it reduces reliance on multicast. In most cases, this will not be possible, in which case the authority MUST advertise addresses that can be used to reach it using mDNS.

4.4. Advertising the Availability of Service Discovery for a Domain

In order for services within an authority dataset to be discovered by DNSSD browsers, the domain that identifies that authority dataset must be advertised as a domain in which discovery should be done. This is accomplished by advertising a PTR record in .local for the legacy browsing domain [RFC6763]. For example, if the domain being used is 'example.wifi.home.arpa.', then the PTR record would be as follows:

```
lb._dns-sd._udp.local IN PTR example.wifi.home.arpa.
```

When more than one authority is advertising discoverability for a particular authority dataset, there will be more than one of these records advertised, but this isn't a problem since they are not required to be unique. This record should not be advertised until the authority has successfully generated or discovered a domain that is unique to the authority dataset being advertised.

5. Discovering Authority Datasets

A DNSSD browser discovers the set of datasets available locally by issuing an mDNS query for lb._dns-sd._udp.local. This query will return zero or more PTR records. As each PTR record is returned, it is compared against the existing set of legacy browsing domains that the DNSSD browser maintains. If the target of the PTR record is not in this set, then it is checked for validity and, if valid, added to the set, and any ongoing browse operations begin to try to browse the new domain.

A browsing domain discovered using mDNS can only be valid if one of the following is true:

- * It is a subdomain of a domain that is defined to be a locally served domain [???
- * The DNSSD browser has found a secure denial of existence for the domain and validated it using DNSSEC
- * The DNSSD browser has found a secure delegation for the domain (in which case it MUST validate answers in that domain using DNSSEC).

In addition, a host on which a DNSSD browser is running may have discovered domains that would be considered valid because it is a locally-served domain, or because it can be proven not to exist in the DNS hierarchy, but for which authoritative service is already provided by the network infrastructure. In this case, the DNSSD browser MUST consider the information provided in multicast DNS to be invalid, and MUST only use the service information provided by the network infrastructure for that domain.

Examples of this would be a domain like 'home.arpa' that is served by the local infrastructure. In this case, if for example the local infrastructure answers with NXDOMAIN for 'example.wifi.home.arpa.' then even if the browser is not able to validate this answer, it MUST treat the mDNS advertisement for this domain as valid, since otherwise the presence of the locally served domain would prevent discovery in mDNS-advertised subdomains even though there is no conflict.

There are three types of browsing domains that might exist in the set of browsing domains maintained by the DNSSD browser.

- * Link-Local: one for each network link to which the DNSSD browser is directly connected
- * Infrastructure: browsing domains provided by the infrastructure
- * Permissionless: browsing domains discovered using mDNS on local links

[RFC6763] and [DNS Push] already describe how to manage link-local and infrastructure browsing domains. Permissionless browsing domains are managed similarly. Each such domain will have one or more name servers. Each name server will, or will not, provide DNS Push service. If one or more servers provide DNS Push service, then the DNSSD browser will, when browsing, attempt to connect to one of the DNS Push servers for that browsing domain, until a successful connection is established or failure is detected. If failure is detected, or if there are no DNS Push servers, the DNSSD browser will use DNS datagrams [RFC1034] to browse that domain.

6. Security Considerations

Multicast DNS provides no mechanism for trust establishment other than the common connection to a shared link. DNSSD browsers are required to treat information about local authority datasets that are advertised using mDNS skeptically. The requirement in section [??] to validate

7. Informative References

Authors' Addresses

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

Email: mellon@fugue.com

Joey Deng
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: deng.qiaoyu@gmail.com

Additional contact information:

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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 25 April 2022

T. Lemon
(L. Qin)
Apple Inc.
22 October 2021

A 'Time Since Registration' Resource Record for Multicast DNS
draft-tllq-tsr-00

Abstract

This document defines a new DNS Resource Record (RR) to be used with multicast DNS. The new RR is used to communicate the time at which the set of RRsets on a domain name were first registered.

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 25 April 2022.

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	2
2. Time Since Registered Resource Record	3
3. mDNS Registrar Behavior	4
4. Internal Handling of TSR records	5
5. Timeliness of Conflict Resolution	5
6. Legacy Registrars	5
7. When to Use TSR	5
8. Registrant API considerations	5
9. Security Considerations	6
10. IANA Considerations	6
11. Informative References	6
12. Normative References	6
Authors' Addresses	7

1. Introduction

Unlike the Internet Domain Name service, with its authority servers and delegation of authority, Multicast DNS has no single source of authority. Because of this, mDNS has a mechanism, conflict resolution Section 9 of [RFC6762], for detecting and fixing conflicts in mDNS advertisements.

The current mechanism for conflict resolution is simple: when a new service is to be advertised, the server that wishes to advertise its service typically registers the service with a central mDNS registrar on the host on which it is running.

This mDNS registrar may have an internal database of services already registered, and may detect a conflict with one of those services. In this case, no network transaction is required: the mDNS registrar immediately detects the conflict and addresses it in one of two ways, depending on what the service requested. The first alternative is that the registrar will report the conflict to the server an error, which the server must fix. Alternatively, the server may have indicated that the mDNS registrar should automatically choose a new name for it, in which case the mDNS registrar does so automatically.

Once any local conflicts have been resolved, the mDNS registrar sends a series of multicast probes to the local network to see if any other host has already registered a service the conflicts with the proposed new service. If such a service is present on the network, the mDNS registrar follows the same process previously described, either reporting the error to the server or automatically choosing a new name.

The effect of this approach is that generally whichever server first registers a service under a particular name wins. If a server comes along later and registers the same service with conflicting information, the newcomer's information is rejected.

This works well for devices acting on their own behalf. However, in the case of advertising proxies, it works poorly: typically an advertising proxy is proxying the contents of its proxy database using mDNS. The source of truth for information in that database is some host that has registered with the proxy, for example using the Service Replication Protocol (SRP).

In the case of an advertising proxy proxying an SRP database, what we want is not the oldest information, but the newest. When the SRP client is able to continue registering with the same SRP server, this works well. However, if SRP is being managed using anycast registration, there is no guarantee that an SRP client will register with the same server each time.

When the SRP client registers with a different server, the behavior we expect with the current conflict resolution approach is that the SRP client will be given a new name, and both the old (stale) advertisement (A) and the new (more recent) advertisement (A') will be seen on the network, as separate services.

This creates a new burden on consumers of that service: they need to parse through the whole list of services of that type, using metadata from the TXT record in the registration if needed to determine that service A and service A' are the same service.

This document proposes an enhancement to the current conflict resolution algorithm for mDNS, which allows an mDNS proxy to report when it received the registration using a new Time Since Registered RR, which is attached to the name of the registration.

2. Time Since Registered Resource Record

The Time Since Registered (TSR) RR is attached to the name for which the TSR RR is asserting a registration time. The TSR RR contains the time in seconds since the most recent registration that has been received. This time is computed at the time that the mDNS message is transmitted, and can be treated by the receiver as relative to the current time.

The resource record is formatted as described in Section 3.2.1 of [RFC1035]. The RDATA consists of the time offset in the form of a 32-bit unsigned number in network byte order.

3. mDNS Registrar Behavior

When probing, an mDNS registrar reports the TSR for the name for which it is probing. When an mDNS Registrar receives a probe, it checks to see if it has any registration that conflicts with the probe announcement. If it does, it compares its internal TSR with the TSR reported in the probe. If the TSR in the probe is more recent than the internal TSR, the internal registration is marked as stale, and the registrar does not respond to the probe. If the TSR in the probe is older than the internal TSR, the registrar reports a conflict as usual.

Note that because TSR computations are affected by network latency, comparisons can't be considered accurate. It is therefore necessary to tolerate some degree of error. As a general rule, a probe containing a TSR that arrives at a registrar for which the timestamp comparison is close to zero should be assumed to be more recent than the registrar's copy: since the registrar already has a registration, that registration most likely arrived before the registration that triggered the probe.

The Service Registration Protocol uses DNS update, and it takes a significant amount of time for a DNS update client to abandon one DNS server for another, so in the absence of significant congestion-related jitter in packet arrival times, it should never be the case that two SRP proxies receive an SRP update at the same time from the same client. Given that SRP generally does not operate across network infrastructure operator boundaries, such delays are unlikely. Also, if such a situation does occur, the updates should contain the same data, and therefore should not be seen by the mDNS registrar as being in conflict.

When a probe succeeds, the registrar that did the probe then announces the new service. Registrars receiving this announcement that have internal registrations that conflict with it, which are marked stale, then remove the internal registration and report this event to the proxy that did the registration.

4. Internal Handling of TSR records

The TSR record that is sent on the wire is expressed in seconds relative to the time of registration. In order to derive a TSR record, the registrar must remember the time at which the registration occurred. This time is recorded as an absolute time, not a relative time. We will refer to it as the TSR timestamp. When sending a TSR RR, the registrar computes the difference between the TSR timestamp, which must always be in the past, and the current system time. This difference is converted to seconds, and that value is then sent as the TSR RR.

5. Timeliness of Conflict Resolution

It is expected that if a conflict exists, it will be recent, and will be resolved quickly. Different systems may be able to record shorter or longer time differences, but because of this expectation of recentness, mDNS registrars should never report a TSR of longer than seven days. It's reasonable to expect that every mDNS implementation should be able to remember time intervals of at least seven days.

6. Legacy Registrars

An mDNS registrar that does not support TSR will treat the TSR record as part of the registration. Since the TSR record is only sent in probes, it will never be erroneously reported to any client that is browsing for services. If a legacy mDNS registrar and an mDNS registrar that supports TSR both advertise the same service, the conflict resolution rules described in RFC6762 will be followed.

7. When to Use TSR

TSR is only relevant for mDNS proxies. It SHOULD NOT be used by regular (non-proxy) mDNS registrants. An mDNS registrant that is a proxy MUST explicitly request that a TSR be used for conflict resolution. mDNS registrars MUST NOT record a TSR timestamp unless the registrant has specifically requested it.

8. Registrant API considerations

When an mDNS proxy registers a service and requests the use of a TSR timestamp, the proxy MUST specify when it received the registration. In order to support this, the API is required not only to allow the registrant to specify that TSR is wanted, but must also provide a way for the proxy to specify an absolute time at which the registration was received.

This is important, for example, in the case of SRP Replication [I-D.lemmon-srp-replication], where an SRP server may receive a registration from a peer during startup synchronization. This registration will have occurred at some significant amount of time in the past, and so it would be incorrect for the mDNS proxy receiving the registration to use the time that the mDNS proxy registers the service as the TSR timestamp.

9. Security Considerations

The TSR RR is an optimization: it ameliorates an edge case for mDNS proxies. A malicious host on the same link could use the TSR RR to win conflict resolution processes. However, because TSR is only used by proxies, this technique will not work for normal mDNS service registrations: in that case, normal mDNS conflict resolution is done, and the attacker gains no benefit from using TSR. In the case of proxied mDNS registrations, an attacker can in fact deny service by superseding existing registrations.

However, such an attacker could achieve the same effect simply by responding to probes with conflict announcements. Furthermore, such an attack would cause noticeable problems on the network which the network operator would then take steps to correct.

Protocols that rely on mDNS MUST NOT assume that mDNS service is secure or private. If security (authentication, authorization and/or secrecy) are needed, these must be provided at the application layer. The use of TSR provides a novel way of attacking some mDNS services, but the ground truth is that if security is not provided at the application layer, this novel attack actually provides no new advantage to the attacker.

10. IANA Considerations

IANA is requested to allocate a new RR Type from the DNS Resource Record (RR) TYPEs registry for the 'Time Since Registered' Resource Record. The type shall be 'TSR'. The value shall be allocated by IANA. The Meaning shall be 'Multicast DNS Time Since Registered'. Reference shall refer to this document, once published. There is no template specified, and IANA shall determine the registration date.

11. Informative References

12. 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>>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.

[I-D.lemon-srp-replication]
Lemon, T., "Automatic Replication of DNS-SD Service Registration Protocol Zones", Work in Progress, Internet-Draft, draft-lemon-srp-replication-00, 26 July 2021, <<https://datatracker.ietf.org/doc/html/draft-lemon-srp-replication-00>>.

Authors' Addresses

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

Email: mellon@fugue.com

Liang Qin
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: Leonqin0101@gmail.com

Additional contact information:

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

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

T. Lemon
(L. Qin)
Apple Inc.
24 April 2022

A 'Time Since Registration' Resource Record for Multicast DNS
draft-tllq-tsr-01

Abstract

This document defines a new DNS Resource Record (RR) to be used with multicast DNS. The new RR is used to communicate the time at which the set of RRsets on a domain name were first registered.

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	2
2. Time Since Registered Resource Record	3
3. mDNS Registrar Behavior	4
4. Internal Handling of TSR records	4
5. Timeliness of Conflict Resolution	5
6. Legacy Registrars	5
7. When to Use TSR	5
8. Registrant API considerations	5
9. Security Considerations	6
10. IANA Considerations	6
11. Informative References	6
12. Normative References	6
Authors' Addresses	7

1. Introduction

Unlike the Internet Domain Name service, with its authority servers and delegation of authority, Multicast DNS has no single source of authority. Because of this, mDNS has a mechanism, conflict resolution Section 9 of [RFC6762], for detecting and fixing conflicts in mDNS advertisements.

The current mechanism for conflict resolution is simple: when a new service is to be advertised, the server that wishes to advertise its service typically registers the service with a central mDNS registrar on the host on which it is running.

This mDNS registrar may have an internal database of services already registered, and may detect a conflict with one of those services. In this case, no network transaction is required: the mDNS registrar immediately detects the conflict and addresses it in one of two ways, depending on what the service requested. The first alternative is that the registrar will report the conflict to the server an error, which the server must fix. Alternatively, the server may have indicated that the mDNS registrar should automatically choose a new name for it, in which case the mDNS registrar does so automatically.

Once any local conflicts have been resolved, the mDNS registrar sends a series of multicast probes to the local network to see if any other host has already registered a service the conflicts with the proposed new service. If such a service is present on the network, the mDNS registrar follows the same process previously described, either reporting the error to the server or automatically choosing a new name.

The effect of this approach is that generally whichever server first registers a service under a particular name wins. If a server comes along later and registers the same service with conflicting information, the newcomer's information is rejected.

This works well for devices acting on their own behalf. However, in the case of advertising proxies, it works poorly: typically an advertising proxy is proxying the contents of its proxy database using mDNS. The source of truth for information in that database is some host that has registered with the proxy, for example using the Service Replication Protocol (SRP).

In the case of an advertising proxy proxying an SRP database, what we want is not the oldest information, but the newest. When the SRP client is able to continue registering with the same SRP server, this works well. However, if SRP is being managed using anycast registration, there is no guarantee that an SRP client will register with the same server each time.

When the SRP client registers with a different server, the behavior we expect with the current conflict resolution approach is that the SRP client will be given a new name, and both the old (stale) advertisement (A) and the new (more recent) advertisement (A') will be seen on the network, as separate services.

This creates a new burden on consumers of that service: they need to parse through the whole list of services of that type, using metadata from the TXT record in the registration if needed to determine that service A and service A' are the same service.

This document proposes an enhancement to the current conflict resolution algorithm for mDNS, which allows an mDNS proxy to report when it received the registration using a new Time Since Registered RR, which is attached to the name of the registration.

2. Time Since Registered Resource Record

The Time Since Registered (TSR) RR is attached to the name for which the TSR RR is asserting a registration time. The TSR RR contains the time in seconds since the most recent registration that has been received. This time is computed at the time that the mDNS message is transmitted, and can be treated by the receiver as relative to the current time.

The resource record is formatted as described in Section 3.2.1 of [RFC1035]. The RDATA consists of the time offset in the form of a 32-bit unsigned number in network byte order.

3. mDNS Registrar Behavior

When probing, an mDNS registrar reports the TSR for the name for which it is probing. When an mDNS Registrar receives a probe, it checks to see if it has any registration that conflicts with the probe announcement. If it does, it compares its internal TSR with the TSR reported in the probe. If the TSR in the probe is more recent than the internal TSR, the internal registration is marked as stale, and the registrar does not respond to the probe. If the TSR in the probe is older than the internal TSR, the registrar reports a conflict as usual.

Note that because TSR computations are affected by network latency, comparisons can't be considered accurate. It is therefore necessary to tolerate some degree of error. As a general rule, a probe containing a TSR that arrives at a registrar for which the timestamp comparison is close to zero should be assumed to be more recent than the registrar's copy: since the registrar already has a registration, that registration most likely arrived before the registration that triggered the probe.

The Service Registration Protocol uses DNS update, and it takes a significant amount of time for a DNS update client to abandon one DNS server for another, so in the absence of significant congestion-related jitter in packet arrival times, it should never be the case that two SRP proxies receive an SRP update at the same time from the same client. Given that SRP generally does not operate across network infrastructure operator boundaries, such delays are unlikely. Also, if such a situation does occur, the updates should contain the same data, and therefore should not be seen by the mDNS registrar as being in conflict.

When a probe succeeds, the registrar that did the probe then announces the new service. Registrars receiving this announcement that have internal registrations that conflict with it, which are marked stale, then remove the internal registration and report this event to the proxy that did the registration.

4. Internal Handling of TSR records

The TSR record that is sent on the wire is expressed in seconds relative to the time of registration. In order to derive a TSR record, the registrar must remember the time at which the registration occurred. This time is recorded as an absolute time, not a relative time. We will refer to it as the TSR timestamp. When sending a TSR RR, the registrar computes the difference between the TSR timestamp, which must always be in the past, and the current system time. This difference is converted to seconds, and that value

is then sent as the TSR RR.

5. Timeliness of Conflict Resolution

It is expected that if a conflict exists, it will be recent, and will be resolved quickly. Different systems may be able to record shorter or longer time differences, but because of this expectation of recentness, mDNS registrars should never report a TSR of longer than seven days. It's reasonable to expect that every mDNS implementation should be able to remember time intervals of at least seven days.

6. Legacy Registrars

An mDNS registrar that does not support TSR will treat the TSR record as part of the registration. Since the TSR record is only sent in probes, it will never be erroneously reported to any client that is browsing for services. If a legacy mDNS registrar and an mDNS registrar that supports TSR both advertise the same service, the conflict resolution rules described in RFC6762 will be followed.

7. When to Use TSR

TSR is only relevant for mDNS proxies. It SHOULD NOT be used by regular (non-proxy) mDNS registrants. An mDNS registrant that is a proxy MUST explicitly request that a TSR be used for conflict resolution. mDNS registrars MUST NOT record a TSR timestamp unless the registrant has specifically requested it.

8. Registrant API considerations

When an mDNS proxy registers a service and requests the use of a TSR timestamp, the proxy MUST specify when it received the registration. In order to support this, the API is required not only to allow the registrant to specify that TSR is wanted, but must also provide a way for the proxy to specify an absolute time at which the registration was received.

This is important, for example, in the case of SRP Replication [I-D.lemon-srp-replication], where an SRP server may receive a registration from a peer during startup synchronization. This registration will have occurred at some significant amount of time in the past, and so it would be incorrect for the mDNS proxy receiving the registration to use the time that the mDNS proxy registers the service as the TSR timestamp.

9. Security Considerations

The TSR RR is an optimization: it ameliorates an edge case for mDNS proxies. A malicious host on the same link could use the TSR RR to win conflict resolution processes. However, because TSR is only used by proxies, this technique will not work for normal mDNS service registrations: in that case, normal mDNS conflict resolution is done, and the attacker gains no benefit from using TSR. In the case of proxied mDNS registrations, an attacker can in fact deny service by superseding existing registrations.

However, such an attacker could achieve the same effect simply by responding to probes with conflict announcements. Furthermore, such an attack would cause noticeable problems on the network which the network operator would then take steps to correct.

Protocols that rely on mDNS MUST NOT assume that mDNS service is secure or private. If security (authentication, authorization and/or secrecy) are needed, these must be provided at the application layer. The use of TSR provides a novel way of attacking some mDNS services, but the ground truth is that if security is not provided at the application layer, this novel attack actually provides no new advantage to the attacker.

10. IANA Considerations

IANA is requested to allocate a new RR Type from the DNS Resource Record (RR) TYPEs registry for the 'Time Since Registered' Resource Record. The type shall be 'TSR'. The value shall be allocated by IANA. The Meaning shall be 'Multicast DNS Time Since Registered'. Reference shall refer to this document, once published. There is no template specified, and IANA shall determine the registration date.

11. Informative References

12. 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>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.
- [I-D.lemon-srp-replication] Lemon, T., "Automatic Replication of DNS-SD Service Registration Protocol Zones", Work in Progress, Internet-

Draft, draft-lemon-srp-replication-01, 7 November 2021,
<<https://datatracker.ietf.org/doc/html/draft-lemon-srp-replication-01>>.

Authors' Addresses

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

Liang Qin
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America
Email: Leonqin0101@gmail.com

Additional contact information:

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