

ANIMA WG
Internet-Draft
Intended status: Standards Track
Expires: 5 September 2022

T.T.E. Eckert
Futurewei
M. Boucadair
C. Jacquenet
Orange
M. Behringer
4 March 2022

DNS-SD Compatible Service Discovery in GeneRic Autonomic Signaling
Protocol (GRASP)
draft-eckert-anima-grasp-dnssd-03

Abstract

DNS Service Discovery (DNS-SD) defines a framework for applications to announce and discover services. This includes service names, service instance names, common parameters for selecting a service instance (weight or priority) as well as other service-specific parameters. For the specific case of autonomic networks, GeneRic Autonomic Signaling Protocol (GRASP) intends to be used for service discovery in addition to the setup of basic connectivity. Reinventing advanced service discovery for GRASP with a similar set of features as DNS-SD would result in duplicated work. To avoid that, this document defines how to use GRASP to announce and discover services relying upon DNS-SD features while maintaining the intended simplicity of GRASP. To that aim, the document defines name discovery and schemes for reusable elements in GRASP objectives.

Note to the RFC Editor

Please replace all occurrences of rfcXXXX with the RFC number assigned to this document.

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 5 September 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. Overview	3
2. Terminology	4
3. Specification	4
3.1. Service and Name Objectives	4
3.2. Objective Value Reuseable Elements Structure	5
3.3. Reuseable Elements	6
3.3.1. Sender Loop Count	6
3.3.2. Service Element	6
3.3.3. Name Element	9
4. Theory of Operation	11
4.1. Using GRASP Service Announcements	11
4.2. Further Comparison with DNS-SD	13
4.3. Open Issues	13
5. Security Considerations	14
6. IANA Considerations	14
7. Acknowledgements	15
8. Contributors	15
9. Change log [RFC Editor: Please remove]	15
9.1. 03 - Refresh	15
9.2. 02 - Revived after charter round 1 finished	15
9.3. 01 -	15
9.4. 00 - Initial version	15
10. References	15
10.1. Normative References	15
10.2. Informative References	16
Authors' Addresses	16

1. Overview

DNS Service Discovery (DNS-SD) [RFC6763] defines a framework for applications to announce and discover services. This includes service names, service instance names, common parameters for selecting a service instance (weight, priority) as well as other service-specific parameters.

GeneRic Autonomic Signaling Protocol (GRASP) [RFC8990] is intended to also be used for service discovery purposes. Reinventing service discovery for GRASP with a similar set of features would result in duplication of work. Therefore, this document defines how to use GRASP to announce and discover services in a way that inherits DNS-SD features and also tries to be compatible in spirit as much as possible while still maintaining the intended simplicity of GRASP.

The goal of this document is to permit defining service and their parameters once and then use that in GRASP, mDNS and (unicast) DNS. Future work can also define DNS-SD <-> GRASP gateway functions.

This document primarily defines how to perform service discovery across such a GRASP domain leveraging GRASP's options to perform unsolicited flooding of announcements or flooding of requests, and finding the closest service instances. Also, the document allows for automatically discovering DNS-SD servers. Such features is meant to optimize the flooding traffic in some deployments.

The initial use case of this document is to support what in DNS-SD is done via mDNS but in larger networks - GRASP-Domains. Beside the efficient flooding, GRASP provides reliability and security, which are depending on the so called substrate used by GRASP for security and hop-by-hop/end-to-end transport, such as the Autonomic control plane (ACP), [RFC8994]. Providing compatibility with existing mDNS service announcer or clients is possible, but not described in this version of the document.

The encoding of information chosen in this document does not try to use GRASP solely as a transport layer, but to also leverage the CBOR structure of GRASP messages to natively encode the message elements required for services in a way that is most simple - instead of using GRASP only as, e.g., an encapsulation of otherwise unchanged DNS message encodings. This is done to minimize the amount of coding required (and not require any DNS code unless future gateway functions are required), to increase the simplicity, minimize the amount of data on the wire, and allow easier extensibility. On the downside, the mechanisms provided here do not cover the whole slew of possible options of DNS/DNS-SD, but instead only those deemed to be required. Others can be added later.

In support of service discovery, this document also defines name discovery and schemes for reusable elements in GRASP objectives which are designed to be extensible so that future work that identifies elements required across multiple objectives do not need to define a scheme how to do this.

2. Terminology

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

This document makes use of terms and concepts defined in [RFC8990].

3. Specification

3.1. Service and Name Objectives

Unsolicited, flooded announcements (M_FLOOD) in GRASP and solicited flooded discovery (M_DISCOVERY) operate on the unit of GRASP technical objectives (identified by 'objective-names' as discussed in Section 2.10 of [RFC8990]). Therefore, a scheme is required to indicate services via 'objective-names'.

Note: Future work may want to reuse the encodings related to services (defined below in this document) inside other (multicast or unicast only) objective exchanges, in which case the service names are not impacted.

When a technical objective (simply referred to as objective) is meant to be solely about a service name, the objective MUST use an 'objective-name' of 'SRV.<service-name>'. This naming scheme is meant to avoid creating duplicates and, potentially, inconsistent name registrations for those objectives vs. registrations done, for example, for DNS-SD.

When an objective is meant announcement and discovery of a DNS compatible <name> such as "www-internal" in "www-internal.example.com", the objective SHOULD use an objective-name of NAME.<name>. See Section 3.3.3 for more details.

3.2. Objective Value Reuseable Elements Structure

Because service discovery, as explained in the prior section, needs to utilize different objectives, it requires cross-objective standardized encoding of the elements of services. GRASP does not define standardized message elements for the message body (called "objective-value") of GRASP messages. Therefore, this document introduces such a feature.

```
objective-value  /= { 1*elements }
elements         /= ( @rfcXXXX: { 1*relement } )

relement  = ( relement-codepoint => relement-value )
relement-codepoint = uint
relement-value     = any
```

If an objective relies upon reusable elements, the 'objective-value' MUST be a CBOR map and the reusable elements are found under the key "@rfcXXXX".

Objectives that do not want reusable elements may use any objective-value format including a CBOR map, but they can not use the "@rfcXXXX" key if they use a map. This approach was chosen as the hopefully least intrusive mechanism given how by nature all of "objective-value" is meant to be defined by individual objective definitions.

The value of "@rfcXXXX" is a map of reusable elements. Each 'relement' has an IANA registered element-name and codepoint (see Section 6). The element-name is for documentation purposes only, CBOR encodings only use the numeric codepoint for encoding efficiency to minimize the risk for this solution to not be applicable to low-bitrate networks such as in IoT.

Format and semantic of the relement-value is determined by the specification of the reusable element as is the fact whether more than one instances of the same reusable element are permitted.

Reusable elements should be defined to be extensible. The methods used depend on the complexity of the element and the likely need to extend/modify the element with backward or non-backward compatible information. The following is a set of initial options to choose from:

Element values that are a map MUST permit and reserve key value 0 (numerical) for private extensions of the element defined by the individual objective.

Element values that are a map MUST NOT use bareword key values starting with a "_". These too are for private extensions defined by the individual objective.

Element values SHOULD be defined so that additional keys in maps and additional elements at the end of arrays can be ignored by prior versions of the definition. Whenever a newer definition is made for an element where this rule is violated, the element SHOULD be changed in a way for older version recipients to recognize that it is not compatible with it.

One method to indicate compatibility is a traditional version "<major>.<minor>". Within the same <major> version number, increasing <minor> version numbers must be backward compatible. Different <major> version numbers are not expected to be compatible with each other. If they are, then this can be indicated by including multiple version numbers.

A compressed form of version compatibility information is the use of a simple bitmask element where each bit indicates a version that the represented data is compatible with.

3.3. Reuseable Elements

3.3.1. Sender Loop Count

```
relement-codepoint /= ( &(sender-loop-count:1) => 1..255 )
```

Sender-loop-count is set by the sender of an objective message to the same value as the loop-count of the message. On receipt, distance = (sender-loop-count - loop-count) is the distance of the sender from the receiver in hops. This element can be used for informational purposes in M_FLOOD and M_DISCOVERY messages and may be required to be used in these messages by the specification of other elements (such as the service element described below). This element MUST occur at most once. If a receiver expects to use the distance but sender-loop-count was not announced, then distance SHOULD be assumed to be 255 by the receiver.

3.3.2. Service Element

The srv-element (service element) is a reusable element to request or announce a service instance or to request and list service instance names.

```
relement-codepoint // = ( &(srv-element:2) => context-element )
```

```
context-element = {
    ?( &(private:0)      => any),
    ?( &(msg-type:1)     => msg-type),
    ?( &(service:2)      => tstr),
    *( &(instance:3)     => tstr),
    ?( &(domain:4)       => tstr),
    ?( &(priority:5)     => 0..65535 ),
    ?( &(weight:6)       => 0..65535 ),
    *( &(kvpairs:7)      => { *(tstr: any) },
    ?( &(range:8)        => 0..255 ),
    *( &(clocator:9)     => clocator),
}
clocator = [ context, locator-option ]
context = cstr
locator-option = ; from GRASP
```

```
msg-type = &( describe: 0, describe-request:1,
               enumerate:2, enumerate-request:3 )
```

Service: A service name registered according to RFC6335. If it is not present, then objective-name MUST be SRV.<service-name> where <service-name> is the service-name.

Instance: The <Instance> of a DNS-SD Service Instance Name (<Instance> . <Service> . <Domain>). It is optional, see Section 4.2.

Domain: The equivalent of the <Domain> field of a DNS-SD Service Instance Name. If domain is not present, this is equivalent to ".local" in DNS (as introduced by mDNS) and implies the unnamed "local" domain, which is the GRASP domain across which the message is transmitted.

Priority, Weight: Service Instance selection criteria as defined in RFC2782. If either one is not present, its value defaults to 0.

Kvpairs: Map of key/value pairs that are service parameters in the same format as the key/value pairs in TXT field(s) of DNS-SD TXT records as defined in RFC6763, section 6.3.

Range: Allows to flexibly combine distance and priority/weight based service selection according to the definition of distance in Section 3.3.1.

If min-distance is the distance of the closest service announcer,

and min-range the range announced by it, then the recipient MUST consider the priority/weight of all service announcers that are not further away than (min-distance + min-range). If not included, range defaults to 255.

If range is announced, the sender-loop-count element MUST also be announced.

Clocator: The "contextual locator" allows to indicate zero or more locators for the indicated service instance. The context element indicates in which context the locator-option is to be resolved. The reserved context value of "" (empty string) indicates the GRASP domain used, aka: the "local" context in which the service announcement is made. The reserved context value of "0" indicates the default routing context of the announcing node. This is often called "global table", "VRF 0" or "default VRF" on nodes using the "VRF" abstraction. Any other value is a string specifying a context such as another VRF.

The mechanism by which originator and recipient of the srv-element agree on common naming for contexts is outside the scope of this specification. The context therefore allows to indicate locators both for the context through which the GRASP message distributed the srv-element (GRASP domain) as well as that for other contexts. Assume the GRASP domain is the ACP, then clocators in ACP would have a context of "", clocators in the global routing table (part of the data-plane) a context of "0", and clocators on other VRFs (also part of data-plane) a clocator that is their string name.

If no locators are indicated, then the locator of the service(s) is the optional locator-option of the GRASP message in which the objective is contained meant to be used for the service(s) indicated and the clocator implied is "".

If locator(s) are indicated, the messages location-option must be ignored for the service (but may be necessary to be present for other purposes of the objective).

Msg-type Type (aka: intention) of the srv-element. If not present, it is assumed to be "describe".

Describe: Describes one service instance. At least one clocator is required for a positive response, all other fields are permitted, but optional. "Describe" is used in M_FLOOD for unsolicited announcements of services (flooded), in M_RESPONSE messages for solicited announcements of a service and in M_NEGOTIATE for negotiated announcements (both unicasted). If clocator is not included, then all fields except service and instance (and msg-

type and private) must not be included and the srv-element provides a negative reply: No information about this service/service instance. This is only permitted in unicasted "describe" messages.

Describe-request: Request for a "describe" reply. It is used in M_DISCOVERY (flooded) for solicited discovery of services or in M_REQ_SYN (unicasted) for negotiated discovery of service instance(s). In "describe-request", only service is mandatory (but can be provided via the objective-name field of the message), and domain is optional. "Instance" is optional. If provided, then the recipient is asked to provide information about the named instance only. All other fields of srv-element are to be ignored by the receiver in this specification, but a semantic for setting them may be introduced in follow-up work, specifically to filter replies by the indicated fields.

"Describe-request" without instance MAY be answered by "Enumerate" (see below) if the responder has so many instances that it thinks the initiator should rather first select one or fewer instances and ask for their description. The sender of the "Describe-request" MUST be prepared to accept that answer and as necessary follow up with "Describe-request" with the instance names of interest.

Enumerate: Used in the same GRASP messages as "describe", but instead of providing information about one service instance, it is listing service instance names. The purpose of enumerate is the same as browsing a service in DNS-SD. It would be followed by some human or automated selection of one or more instances and then a "describe" M_REQ_SYN request for those instances sent to the source of the "enumerate" to learn about the locators and other parameters of the service instances.

In this specification, all fields other than service, instance and domain (and msg-type and private) must be unset in "enumerate".

Enumerate-request: Requests an "enumerate" reply. It is used in the same way as "Describe-request" except that instance would usually not be set (because in that case it is more useful to send a "Describe-request").

3.3.3. Name Element

The NAME,<name> elements is meant to provide basic name resolution comparable to mDNS name resolution for GRASP domains where this is desirable and no better name resolution exist - for example in the ACP where there is no requirement for DNS.

Because the GRASP service lookup (unlike) DNS does not mandate that nodes have names (not even service instance names), the use of names is primarily meant to support legacy software. New designs should instead look up only services and service instance names, and nodes should announce their names as service instance names for the services they offer:

For example consider a GRASP (ACP) domain of "example.com". The node providing some "www" service could have a name "www-internal" which means GRASP objective NAME.www-internal, that objective value would include primarily the nodes IP address(es) and the port number for the www service would have to be guessed (80). Better, the node would announce GRASP objective SRV.www and the objective value would include the service instance name www-internal and the (TCP) port information (80 or a non-default port).

```
relement-codepoint //= ( &(name-element:3) => context-element )
```

```
context-element //= {
    *( &name:10)          => tstr),
}
```

```
ipv6-address-option = [O_IPv4_ADDRESS, ipv6-address]
ipv4-address-option = [O_IPv6_ADDRESS, ipv6-address]
locator-option /= ipv4-address-option
locator-option /= ipv6-address-option
```

Name information is carried in the name-element relement. It is a context-element like the one used for srv-element except that it adds the name component and that it does not permit the service and instance components and that it allows only describe and describe-request values in the msg-type. Clocators MUST use the ipv6-address-option or ipv4-address-option in the locator-option component.

TBD: Unclear if/how we should best formalize the differences in the context element permitted information between services and names. The above is quite informal.

Priority, weight, kvpairs, range (and of course private) MAY be used in describe messages to support multiple instances of the same name, as used for name anycast/prioritycast.

Nodes may have multiple names. These can be listed in the name component. If a nodes names have the notion of a primary name and secondary names then the primary name should be the first in the list of names. In DNS-SD, the name pointed to by CNAME RRs can be considered to be the primary name. A describe-request for a non-primary name SHOULD return in the list of names the requested name and the primary name.

Note that there is no reverse lookup defined in this version of the document (no lookup from IP address to name).

4. Theory of Operation

4.1. Using GRASP Service Announcements

TBD: This section contains a range of details that should become normative in later versions.

This section provides a step by step walk-through of how to use GRASP service announcements and compares it to DNS-SD.

The most simple method to use GRASP service discovery is to select (and if still necessary, register) a <service-name> and start one or more agents (e.g.: ASAs) announcing their service instance(s) via GRASP. At minimum, an agent should periodically (default 60 seconds) announce the service instance via GRASP M_FLOOD messages as an objective SRV.<service-name> with a srv-element and a sender-loop-count element (default 255). The ttl of the GRASP message should be 3.5 times the announcement period, e.g.: 210000 msec.

Consumers of the service will use GRASP to learn of the service instances and select one. This approach is most similar to the use of DNS-SD with mDNS except that the scope of the announcement is a whole GRASP domain (such as the ACP) as opposed to a single IP subnet in mDNS and that mDNS primarily relies on request & reply but in its standard not on periodic unsolicited announcements. We describe here the unsolicited flooding option via M_FLOOD first because it is recommended for services with a dense population of service consumers and it is most simple to describe.

On the service announcer, the parameters priority, weight and range of the service instance can be selected from intent or configuration - or left at default. The default range 255 will result in selection of a random target of the service like in DNS-SD. Setting priority/weight allows to prioritize and weigh the selection as in DNS-SD. Setting range to 0 allows to select the closest target, priority/weight are only compared between targets of the same shortest

distance. Distance based options are not available in DNS-SD because it does not expect that network distance is available to arbitrary DNS-SD client. It is available to GRASP clients though. Using 0 < range < 255 allows for a hybrid priority/weight and distance based service selection (e.g.: Select the highest priority instance within a range of 5 hops).

If the service is a non-GRASP service, then the result of the service discovery has to be a transport locator to which the client can open a connection and talk the protocol implied by the service. This transport locator(s) have to be put into the clocator parameter. The context of the clocator would normally be "", aka: the transport locator is in the IP reachability associated with the GRASP domain (e.g.: IPv6 of the ACP for ACP GRASP domain).

If an ACP service is announced via ACP GRASP, then the locator(s) can be O_IPv6_LOCATOR or O_FQDN_LOCATOR. The O_IPv6_LOCATOR is used if the service is defined to be available via some transport layer port (TCP, UDP or other). The determination of the actual transport connection to be used is the same as in DNS-SD: If the transport protocol is not TCP or UDP, it has to be implied by the specification of <service-name> or can be detailed in kvpairs which carries the same information as DNS-TXT TXT RRs of the service. Alternatively, the transport-proto field of the locator can contain any valid IP protocol directly (TBD), which is not possible in DNS-SD.

Like DNS-SD, service discovery via GRASP does not require allocation and use of well-known ports for services. Unlike DNS-SD, there is no need in GRASP to define service instance names or target names. In DNS SD, PTR RRs resolve from a service name to a set of service instance named. SRV and TXT RRs resolve from service instance names to service instance parameters including the target. A target is the DNS host name of the service instance. It gets resolved via A/AAAA RRs to IPv4/IPv6 addresses of the target. In GRASP service discovery, host names are not used. Service instance names are optional too. Service instance names are useful for human diagnostics and human selection of service instances. In fully automated environments, they can be are less important. For diagnostic purposes, it is recommended to give service instances service instance names in GRASP service announcements.

A locator with O_URI_LOCATOR type can be used in GRASP to indicate a URI for the transport method for a service instance. If the URI includes a host part, care must be taken to use only IP addresses in the host part if the context of the GRASP domain does not support host name resolution - such as the ACP - or to use the GRASP name resolution mechanisms described elsewhere in this document. And that the addresses indicated are also reachable in the GRASP domain. For

example, in service announcements across a DULL GRASP domain, only the IPv6 link-local addresses on that subnet must be used (this applies equally when using the O_IPv6_LOCATOR).

Instead of using M_FLOOD to periodically announce service instances, M_DISCOVERY can be used to actively query for service instances. The msg-type type must then be "describe-request". Because no periodic flooding is necessary, this solution is more lightweight for the network when the number of requesting clients is small. Note though that the M_DISCOVERY will terminate as soon as a provider of the objective is found, so the service instances found will be based on distance and therefore selection of instance by priority and weight will not work equally well as with M_FLOOD. Consider for example a central service instance in the NOC that should always be used (for example for centralized operational diagnostics) unless the WAN connection is broken, in which case distributed backup service instances should be used. With the current logic of M_DISCOVERY this is not possible.

4.2. Further Comparison with DNS-SD

Neither the GRASP SRV.* objective-name, the service name nor any other parameter explicitly indicate the second label "_tcp" or "_udp" of DNS-SD entries. DNS-SD, RFC6763 explains how this is an unnecessary, historic artifact.

This version of the document does not define an equivalent to "_sub" structuring of service enumeration.

This version of the document does not define mechanisms for reverse resolution of arbitrary services: An inquirer may unicast M_SYNC_REC to a node with a series of objectives with specific service names of interest and describe-request, but there is no indication of "ANY" service.

4.3. Open Issues

TBD: Examine limitations mentioned in "in this version of the text/document".

TBD: The GRASP specification does currently only permit TCP and UDP for the transport-proto element. This draft should expand the GRASP definitions to permit any valid IP protocol. We just need to decide whether this should only apply to the locator in the srv element or also retroactive to the locator-option in GRASP messages (maybe not there ?).

TBD: A fitting CBOR representation for a kvpair key without value needs to be specified so that it can be distinguished from an empty value as outlined in RFC6763 section 6.4.

TBD: In this version, every service/service-instance is an element by itself. Future versions of this document may add more encoding options to allow more compact encoding of recurring fields.

TBD: Is there a way in CDDL to formally define the string names of the relement-codepoint's ?

5. Security Considerations

TBD.

GRASP-related security issues are discussed in Section 3 of [RFC8990].

6. IANA Considerations

This document requests IANA to create a new "GRASP Objective Value Standard Elements" subregistry under the "GeneRic Autonomic Signaling Protocol (GRASP) Parameters" registry.

The values in this table are names and a unique numerical value assigned to each name. Future values MUST be assigned using the RFC Required policy as dedfined in Section 4.7 of [RFC8126]. The numerical value is simply to be assigned sequentially. The following initial values are assigned by this document:

sender-loop-count 1 [defined in rfcXXXX]

srv-element 2 [defined in rfcXXXX]

name-element 3 [defined in rfcXXXX]

This document updates the handling of the "GRASP Objective Names" Table introduced in the GRASP IANA considerations as follows:

Assignments for objective-names of the form "SRV.<text>" and "NAME.<text>" are special.

Assignment of "SRV.<text>" can only be requested if <text> is also a registered service-name according to RFC6335. The specification required for registration of a "GRASP Objective Name" MUST declare that the intended use of the objective name in GRASP is intended to be compatible with the indented use of the registered service name.

Registration of "SRV.<text>" in the "GRASP Objective Name" table is optional, but recommended for all new service-names that are meant to be used with GRASP. Non-registration can for example happen with DNS-SD <-> GRASP gateways that inject pre-existing service-names into GRASP. Note that according to the GRASP RFC, registration is mandatory, so this exemption for "SRV.<text>" is also an update to that specification.

There MUST NOT be any assignment for objective names of the form "NAME.<text>". These names are simply used by GRASP nodes without registration (just like names in mDNS).

7. Acknowledgements

8. Contributors

Brian Carpenter

9. Change log [RFC Editor: Please remove]

9.1. 03 - Refresh

9.2. 02 - Revived after charter round 1 finished

Reviving after ANIMA charter 01 is finished, adding new co-authors, contributors.

Textual improvements, updating references.

9.3. 01 -

Only refreshing, no changes since -00.

9.4. 00 - Initial version

10. References

10.1. 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>>.
- [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>>.

- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [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>>.
- [RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <<https://www.rfc-editor.org/info/rfc8990>>.

10.2. Informative References

- [RFC8994] Eckert, T., Ed., Behringer, M., Ed., and S. Bjarnason, "An Autonomic Control Plane (ACP)", RFC 8994, DOI 10.17487/RFC8994, May 2021, <<https://www.rfc-editor.org/info/rfc8994>>.

Authors' Addresses

Toerless Eckert
Futurewei Technologies USA Inc.
2220 Central Expressway
Santa Clara, 95050
United States of America
Email: tte+ietf@cs.fau.de

Mohamed Boucadair
Orange
35000 Rennes
France
Email: mohamed.boucadair@orange.com

Christian Jacquenet
Orange
35000 Rennes
France
Email: christian.jacquenet@orange.com

Michael H. Behringer
Email: michael.h.behringer@gmail.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

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