SMPTE STANDARD

Media Device Control Discovery (MDCD)

Table of Contents

Approved
May 13, 2014

Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standards developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices, and Engineering Guidelines, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in of its Standards Operation Manual. SMPTE ST 2071-3 was prepared by Technology Committee 34CS.

Intellectual Property

At the time of publication no notice had been received by SMPTE claiming patent rights essential to the implementation of this Engineering Document. However, attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. SMPTE shall not be held responsible for identifying any or all such patent rights.

Introduction

Since the inception of media devices there has been a continual need for a standardized means of controlling them. This need led to the initial creation of protocols such as de-facto, manufacturer based, serial RS-422 control on a 9-pin "D" connector, and later Video Disk Control Protocol (VDCP). However, unfortunately as technologies advanced and media devices became connected to Internet Protocol networks these methods were replaced with proprietary solutions. This Media Device Control suite of standards is intended to address this issue and deliver the same level of interoperability as its predecessors, using Internet Protocol, with provisions for extensibility and adaptability. This document describes the provisions necessary for network-based discovery and Zero Configuration (ZeroConf), allowing MDC compliant devices to register themselves to the system and to locate other MDC devices, with little to no configuration. The provisions in this document are designed to scale from small, unmanaged networks to large enterprise networks and to the Internet, providing mechanisms to simplify configuration wherever possible.

The following diagrams depict two MDC network configurations where MDC devices configure themselves to discover other MDC devices with little to no configuration, beyond that which is needed to access the network. The first diagram (Figure 1) depicts a network where MDC devices discover one another utilizing DHCP and DNS. The second diagram (Figure 2) illustrates how a MDC compliant device and a MDC client can be connected, with a single network cable, and automatically configure themselves into a functioning Link-Local network, where the client can discover the device and interrogate the device for its capabilities.

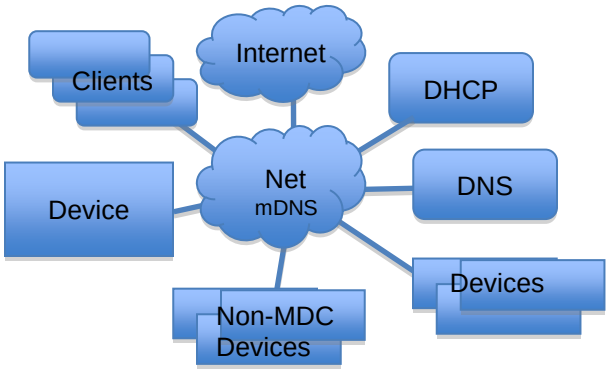

Figure 1 – Device configuration via DHCP and DNS

Figure 2 – Two directly connected Devices using Link-Local

1 Scope

This document describes the Zero Configuration (ZeroConf) and service discovery mechanisms defined for the Media Device Control (MDC) [SMPTE ST 2071] suite of standards. Defining how MDC compliant devices utilize existing network services, or in the absence of such services, use existing Zero Configuration (ZeroConf) techniques and standards to automatically construct a fully functional system.

Design Goals

- Allow for the querying of devices and services within logical discovery domains.
- Allow for the separation of the discovery and primary name resolution infrastructures.
- Allow clients and devices to work around hardware and network failures, through discovery.
- Reduce and centralize the configuration required to create a functional Media Device Control system.
- Use common, well understood, and widely adopted standards and technologies.

2 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any section explicitly labeled as "Informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword “reserved” indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword “forbidden” indicates “reserved” and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

Unless otherwise specified, the order of precedence of the types of normative information in this document shall be as follows: Normative prose shall be the authoritative definition;

Tables shall be next; followed by formal languages; then figures; and then any other language forms.

3 Normative References

The following standards contain provisions that, through reference in this text, constitute provisions of this recommended practice. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this recommended practice are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

IETF RFC 1034, Domain Names – Concepts and Facilities

IETF RFC 1035, Domain Names – Implementation and Specification

IETF RFC 1464, Using the Domain Name System to Store Arbitrary String Attributes

IETF RFC 2131, Dynamic Host Configuration Protocol (DHCP)

IETF RFC 2136, Dynamic Updates in the Domain Name System (DNS UPDATE)

IETF RFC 2181, Clarifications to the DNS Specification

IETF RFC 2782, A DNS RR for specifying the location of services (DNS SRV)

IETF RFC 3007, Secure Domain Name System (DNS) Dynamic Updates

IETF RFC 3315, Dynamic Host Configuration Protocol for IPv6 (DHCPv6)

IETF RFC 3927, Dynamic Configuration of IPv4 Link-Local Addresses (IPv4LL)

IETF RFC 4034, Resource Records for the DNS Security Extensions

IETF RFC 4862, IPv6 Stateless Address Autoconfiguration

IETF RFC 6106, IPv6 Router Advertisement Options for DNS Configuration

IETF RFC 6762, Multicast DNS (mDNS).

IETF RFC 6763, DNS-Based Service Discovery (DNS-SD)

SMPTE ST 2071-1:2014, Media Device Control Framework (MDCF)

SMPTE ST 2071-2:2014, Media Device Control Protocol (MDCP)

4 Name Resolution (Informative)

Name resolution is the process of resolving human readable names into protocol specific addresses. Name resolution can be provided in a centralized fashion, as a service, such as Domain Name System (DNS), or it can be provided as a decentralized broadcast based network protocol, such as Multicast DNS (mDNS) or Link-Local Multicast Name Resolution (LLMNR). If name resolution is implemented as a service, the resolver sends a unicast request to the name resolution service and awaits a unicast response. If name resolution is implemented as a broadcast based network protocol, a querier broadcasts the name resolution request to a broadcast or multicast address and awaits responses from any services that may reply. In either methodology, one or more systems are configured to accept name resolution requests and to respond to those requests with the appropriate protocol specific addresses. Name resolution may also be used in reverse, to find the human readable name for a protocol address in a process known as reverse mapping.

The Domain Name System (DNS) ([RFC 1034], [RFC 1035], [RFC 4795], and [RFC 6762]) is the standard name resolution service defined for Internet Protocol networks. At the time of this writing, the IETF defines three variants of DNS, one using a unicast service model, called Domain Name System (DNS) ([RFC 1034] and [RFC 1035]) and two broadcast network protocol variants, using multicast UDP datagrams, known as Link-Local Multicast Name Resolution (LLMNR) [RFC 4795] and Multicast DNS (mDNS) [RFC 6762]. When using DNS, a client, known as a resolver, issues a unicast UDP or TCP name resolution request to the DNS name server. The DNS name server replies to the requestor in a unicast fashion, using either UDP or TCP, depending upon the protocol that was used for the request. When using LLMNR, the requestor broadcasts a request using a multicast UDP datagram and like DNS, the LLMNR service replies to the requestor in a unicast fashion. Multicast DNS (mDNS) is similar to LLMNR. However, the mDNS service, known as a querier, replies to the requestor in a multicast fashion, using multicast UDP datagrams. The multicast nature of the response allows all mDNS resolvers on the Link-Local network to receive and cache the name resolution responses for each request. This pre-caching allows mDNS resolvers to resolve future name resolution requests without generating network traffic. Both LLMNR and mDNS use the same data format as DNS for request and response data packets and the packets can be confused between the DNS variants. Therefore, each DNS variant is assigned a distinct default IP port number to prevent systemic issues caused by the minor differences between the DNS, LLMNR, and mDNS datagrams.

4.1 Domain Name System (DNS) (Normative)

The Domain Name System (DNS) ([RFC 1034] and [RFC 1035]), the centralized, service-based name resolution service, shall be used to implement MDC Discovery. Domain Name System is the standard name resolution service defined for Internet Protocol networks. The use of DNS allows MDCD to be implemented using existing, well defined, and well-understood technologies that are usually already present on the network. Please refer to Annex B DNS Primer (Informative) for further details regarding the Domain Name System (DNS).

4.2 Multicast DNS (mDNS) (Normative)

Multicast DNS (mDNS) [RFC 6762], the decentralized, broadcast based network protocol for name resolution, shall be used to implement MDC Discovery (MDCD). Multicast DNS (mDNS) is an extension of DNS that uses multicast UDP to broadcast name resolution requests and responses over the Link-Local network. mDNS packets are nearly identical to DNS packets; however, extensions have been added to allow for a single mDNS packet to carry multiple name resolution requests and responses. Unlike LLMNR and DNS, mDNS queriers broadcast all name resolution responses and may receive unsolicited name resolution responses. mDNS queriers will cache name resolution responses, whether requested or not, for future use and to reduce the network traffic generated by service discovery.

In later sections of this document, normative algorithms and processes are defined that use DNS and mDNS for MDC Discovery. Within these algorithms the “local” and reverse mapping Link-Local domains will utilize mDNS for name resolution. All domains that are not designated as mDNS domains shall use unicast DNS for name resolution. The complete list of mDNS domains is listed in Table 1.

Table 1 – Multicast DNS (mDNS) Domains

Internet Protocol Version	Multicast DNS Domain
IPv4 and IPv6	local
IPv4	254.169.in-addr.arpa
IPv6	8.e.f.ip6.arpa
IPv6	9.e.f.ip6.arpa
IPv6	a.e.f.ip6.arpa
IPv6	b.e.f.ip6.arpa

5 DNS-Based Service Discovery (DNS-SD)

MDC Discovery shall be an extension of DNS-Based Service Discovery (DNS-SD) [RFC 6763] with additional steps defined to allow for the use of DNS name servers that are not part of the network's primary naming infrastructure. DNS-Based Service Discovery (DNS-SD) allows MDC Discovery to operate using well-understood, mature technologies that have numerous implementations and a wide variety of available resources, such as books, white papers, third party libraries, coding examples, and human resources. The use of DNS further allows MDC Discovery to operate within an Internet Protocol network without the need for additional equipment, services, or expertise; providing scalability from two directly connected devices up to the Internet and everything in between.

MDC Discovery processes may use DNS servers that are not part of the network's primary DNS infrastructure by using domain hierarchies or naming services that are defined as DNS services, using the DNS service type "_domain._udp". Please refer to Section 8.4.3, Finding Additional Name Servers, for further details.

6 Devices and DNS-Based Service Discovery (DNS-SD)

Each MDC Discovery compliant device shall expose its Device Capability Interface as a DNS-SD compliant service instance. Each DNS-SD service instance shall be represented as a DNS SRV Resource Record (SRV RR) and a corresponding DNS TXT Resource Record (TXT RR), as defined by DNS-SD [RFC 6763]. The TXT RR associated to the SRV RR shall be formatted to the specification in Section 7.3, Text (TXT) Resource Record, and there shall be two DNS PTR Resource Records (PTR RR) defining the service type and service subtype pointers that point to the SRV and TXT RRs. The first PTR RR shall be of the name `_mdc._tcp`, while the second PTR RR shall be named in accordance to the specification in Section 6.1, Capability Interfaces and Service Subtypes.

Consequently, each MDC Discovery compliant device shall have an equivalent of the following DNS configuration specified within the DNS or mDNS name resolution infrastructures.

```
device1                             IN A 192.168.0.21
_mdc._tcp                           IN PTR Instance._device_v1._sub._mdc._tcp
_device_v1._sub._mdc._tcp           IN PTR Instance._device_v1._sub._mdc._tcp
Instance._device_v1._sub._mdc._tcp  IN SRV 10 10 8080 device1
    TXT ("txtvers=1" "rn=urn:smpte:udn:namespace1:device1" "proto=mdcp" "path=/MDC/Device")
```

6.1 Capability Interfaces and Service Subtypes

Any Capability Interface implemented by a device should be made discoverable via MDC Discovery. As discussed in SMPTE ST 2071-1, each Capability Interface is assigned a unique UCN and each UCN contains the interface name, version, and the interface namespace of the Capability Interface. To translate a UCN into an MDC Discovery compliant service subtype, the initial text "urn:smpte:ucn:" shall be stripped from the UCN and an underscore prepended to the remaining text. Then the text "`._sub._mdc._tcp`" shall be appended to the end of the text, giving the equivalent text:

_[*<interface namespace>*:]*<interface name>*_v*<version>*._sub._mdc._tcp

Where the interface namespace is optional and dependent upon its presence in the original UCN.

The format of the Capability Interface's service subtype shall be in the form:

```
"_" (UCN – "urn:smpte:ucn:") "._sub._mdc._tcp"
```

or when expressed as psuedocode

```
UCN.replace("urn:smpte:ucn:", "_") + "._sub._mdc._tcp"
```

Examples:

```
urn:smpte:ucn:device_v1 → _device_v1._sub._mdc._tcp
urn:smpte:ucn:device_directory_v1 → _device_directory_v1._sub._mdc._tcp
urn:smpte:ucn:media_directory_v1 → _media_directory_v1._sub._mdc._tcp
urn:smpte:ucn:some company:iface_v1.0.0 → _some company:iface_v1.0.0._sub._mdc._tcp
```

7 DNS Resource Records for MDC Discovery

MDC Discovery shall be based on DNS-Based Service Discovery (DNS-SD), using 3 Resource Record types (PTR, SRV, and TXT), in addition to the standard hostname to IP Address translation Resource Record types (A and AAAA). This section specifies the definition of the DNS-SD PTR, SRV, and TXT Resource Records for MDC Discovery and defines the attributes that shall be carried within the MDC Discovery TXT Resource Record associated to the SRV Resource Record that represents the Capability Interface endpoint.

Please note that the DNS RR configuration examples in this section use the standard "BIND" configuration format. "BIND" is the most well-known and widely adopted DNS name server implementation used throughout the Internet and therefore the "BIND" configuration format is a well-known method of depicting DNS RRs.

7.1 Pointer (PTR) Resource Record

Pointer (PTR) Resource Records ([RFC 1035] and [RFC 2181]) shall be used to associate one service type to one or more service instances. Multiple PTR records may share the same service type, but each shall contain a unique combination of service type and service instance.

7.1.1 Pointer (PTR) Resource Record Format

The format of the PTR Resource Record ([RFC 1035] and [RFC 2181]) shall be in the form:

```
Service TTL Class "PTR" Instance.Service
```

`Service` The symbolic name of the service type or subtype. Shall conform to the naming convention established for service and service subtype names specified in Section 6.1, Capability Interfaces and Service Subtypes.

`TTL` Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the number of seconds that this RR can be cached before the information should be re-queried from the source. Note that a value of '0' indicates that the RR should not be cached and shall be discarded immediately. (Optional: Defaults to the Zone TTL)

`Class` Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to "IN", without the quotes, if unspecified. (Optional)

`"PTR"` Indicates that this RR is a PTR RR. The value shall be equal to "`PTR`", without the quotes.

`Instance` Shall contain human readable, descriptive name of the service instance to which this PTR RR pertains. Please refer to the DNS-SD specification [RFC 6763] for more details.

Configuration Examples:

```
_mdc._tcp                                IN PTR Instance._device_v1._sub._mdc._tcp
_device_v1._sub._mdc._tcp                IN PTR Instance._device_v1._sub._mdc._tcp
_device_directory_v1._sub._mdc._tcp      IN PTR Instance._device_directory_v1._sub._mdc._tcp
_media_directory_v1._sub._mdc._tcp       IN PTR Instance._media_directory_v1._sub._mdc._tcp
```

Where the service instances are named simple “Instance”.

7.2 Service (SRV) Resource Record

Service (SRV) Resource Records [RFC 2782] shall be used to specify the network location of a service within the domain. Each device shall have one SRV RR for each Capability Interface exposed by MDC Discovery.

7.2.1 Service (SRV) Resource Record Format

The format of the SRV Resource Record [RFC 2782] for MDC Discovery shall be in the form:

```
Instance.Service TTL Class SRV Priority Weight Port Target
```

`Instance` Shall contain human readable, descriptive name of the service instance to which this TXT Resource Record pertains. Please refer to the DNS-SD [RFC 6763] specification for details.

`Service` The symbolic name of the service type. Shall conform to the naming convention established for service subtype names specified in Section 6.1 - Capability Interfaces and Service Subtypes.

`TTL` Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the number of seconds that this RR can be cached before the information should be re-queried from the source. Note that a value of ‘0’ indicates that the RR should not be cached and shall be discarded immediately. (Optional: Defaults to the Zone TTL)

`Class` Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to “IN”, without the quotes, if unspecified. (Optional)

`"SRV"` Indicates that the RR is a Service RR The value shall be equal to ”SRV”, without the quotes.

`Priority` Shall contain the Priority of this Resource Record relative to other SRV Resource Records sharing the same service instance name. Zero (0) is the highest priority.

`Weight` Shall specify the weight of the SRV record relative to other Resource Records sharing the same service instance name, with the same Priority.

`Port` Shall contain the value of the Internet Protocol Port used by the endpoint on the Target host.

`Target` Shall contain the domain name of target host.

Configuration Examples:

```
Instance._device_v1._sub._mdc._tcp              IN SRV 10 10 8080 device1
Instance._device_directory_v1._sub._mdc._tcp    IN SRV 10 10 8080 device1
Instance._media_directory_v1._sub._mdc._tcp     IN SRV 10 10 8080 device1
```

Where the service instances are named simple “Instance” and are provided by the host named “device1”.

7.3 Text (TXT) Resource Record

Text (TXT) Resource Records ([RFC 1035] and [RFC 1464]) shall be used to associate textual attributes to each SRV Resource Record. As defined by the DNS-SD specification [RFC 6763], each MDC Discovery SRV Resource Record shall have a corresponding TXT Resource Record, even if that TXT RR is empty.

7.3.1 Text (TXT) Resource Record Format

The format of the TXT Resource Record [RFC 1464] shall be in the form:

`Instance.Service Class TTL "TXT" ( "AName=AValue" "AName=AValue" … )`

`Instance` Shall contain human readable, descriptive name of the service instance to which this TXT Resource Record pertains. Please refer to the DNS-SD [RFC 6763] specification for details.

`Service` The symbolic name of the service type. Shall conform to the naming convention established for service subtype names specified in Section 6.1, Capability Interfaces and Service Subtypes.

`TTL` Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the number of seconds that this RR can be cached before the information should be re-queried from the source. Note that a value of '0' indicates that the RR should not be cached and shall be discarded immediately. (Optional: Defaults to the Zone TTL)

`Class` Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to "IN", without the quotes, if unspecified. (Optional)

`"TXT"` Indicates that this RR is a Text RR. The value shall be equal to "TXT", without the quotes.

`AName` Shall contain the unique attribute name.

`AValue` Shall contain the value of the attribute named in the `AName` field.

Configuration Examples:

```
Instance._device_v1._sub._mdc._tcp        IN SRV 10 10 8080 device1
    TXT ("txtvers=1" "rn=urn:smpte:udn:namespace1:device1" "proto=mdcp" "path=/MDC/Device")
```

Where the service instances are named simple "Instance" and four attributes are defined to provide the required information to connect to a single service exposed by "device1" on port 8080.

7.3.2 Text (TXT) Resource Record Attributes for MDC Discovery

Each MDC Discovery TXT Resource Record (TXT RR) shall specify the Resource Name (RN), wire level protocol, and URL path by which the Capability Interface endpoint will be accessed and may also specify the format version used by this TXT RR. The attributes from the TXT RR will be used in conjunction with the information from the corresponding SRV RR to construct the full URL needed to access the Capability Interface endpoint and will indicate the data protocol used for message exchange between the client and the endpoint.

7.3.2.1 "txtvers" – Text Resource Record (TXT RR) Version

The `txtvers` attribute shall contain the version of the MDC Discovery TXT RR format. The `txtvers` attribute is included to help clients maintain backward compatibility and if present shall contain the value "1", without the quotes, to indicate the TXT RR format described within this document section. If the `txtvers` attribute is not present, the default value shall be "1".

7.3.2.2 "rn" – Resource Name (RN)

The `rn`attribute shall specify the unique RN that identifies the device or service that exposes the Capability Interface represented by this SRV / TXT RR pair. All MDC Discovery TXT RRs shall contain a`rn`attribute.

7.3.2.3 "proto" – Wire Protocol for Capability Interface

The `proto` attribute shall specify the wire level protocol used to exchange messages between the client and the Capability Interface endpoint, represented by this SRV / TXT RR pair. Table 2 indicates the acceptable values for the `proto` attribute and the corresponding wire level protocol each represents. All MDC Discovery TXT RRs shall contain a `proto` attribute.

Table 2 – Protocols Cross Reference

Value of "proto" Attribute	URL Protocol To Use	Wire Protocol
mdcp	http	ST2071-2
soap_bp11	http	WS-I BP 1.1
soap_bp12	http	WS-I BP 1.2
soap_bp20	http	WS-I BP 2.0

Please note that WS-I BP 1.1, WS-I BP 1.2, and WS-I BP 2.0 refer to the OASIS WS-I Basic Profiles 1.1, 1.2, and 2.0, respectively. New `proto` values may be added in the future and vendors may define their own proprietary `proto` values for additional wire level protocols.

7.3.2.4 "path" – Path Component of the URL for the Capability Interface Endpoint

The `path` attribute shall specify the path portion of the URL used to access the Capability Interface endpoint represented by this SRV / TXT RR pair. All MDC Discovery TXT RRs shall contain a `path` attribute.

7.4 Next-Secure (NSEC) Resource Record

Next-Secure (NSEC) Resource Records [RFC 4034] should be used to indicate authoritative Resource Records present for a PTR RR. NSEC RRs contain two separate values, the Next Domain Name, in the canonical ordering of the zone, and the set of RR types present for the NSEC records Next Domain Name. For example the Next Domain Name could be the hostname associated to a PTR RR and a RRSet indicating the authoritative RR types that are available for the Next Domain Name, such as SRV, TXT, A, AAAA, etc.

7.4.1 Next-Secure (NSEC) Resource Record Format

The format of the NSEC Resource Record ([RFC 1035] and [RFC 2181]) shall be in the form:

`Service TTL Class "NSEC" NDName RRSet`

`Service` The symbolic name of the service type or subtype. Shall conform to the naming convention established for service and service subtype names specified in Section 6.1, Capability Interfaces and Service Subtypes.

`TTL` Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the number of seconds that this RR can be cached before the information should be re-queried from the source. Note that a value of '0' indicates that the RR should not be cached and shall be discarded immediately. (Optional: Defaults to the Zone TTL)

`Class` Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to "IN", without the quotes, if unspecified. (Optional)

`"NSEC"` Indicates that this RR is a NSEC RR. The value shall be equal to "`NSEC`", without the quotes.

`NDName` Shall contain the Next Domain Name that has authoritative data for the PTR RR associated to the name specified in the `Service` field.

`RRSet` Shall contain the list of authoritative RR types associated to the `NDName`.

Configuration Examples:

```
Instance._device_v1._sub._mdc._tcp  IN   NSEC   device1 TXT SRV
```

Where the service instance is named simply “Instance” and only authoritative TXT and SRV RRs exist for the hostname “device1”.

7.5 Constructing URLs from the SRV and TXT Resource Records

Each SRV / TXT RR pair provides all of the information required to construct the full URL needed to access and exchange messages with the Capability Interface endpoint. The URL shall be constructed as follows:

```
PROTOCOL = Wire Protocol from Table 1, Protocols Cross Reference, for the TXT.proto attribute
PROTOCOL "://" SRV.Target ":" SRV.Port TXT.path
```

For example, the following SRV / TXT RR pair would produce the URL `http://device1:8080/MDC/Device`

```
System._device_v1._sub._mdc._tcp          IN SRV 10 10 8080device1
    TXT ("txtvers=1" "rn=urn:smpte:udn:namespace1:device1" "proto=mdcp" "path=/MDC/Device")
```

8 Zero Configuration

Zero Configuration (ZeroConf) is the process of automatically configuring network hosts without the need for pre-configuration. ZeroConf is implemented using a number of well known, widely adopted, standards that automatically allocate network addresses, provide name resolution services, and facilitate the discovery of network based services. This section specifies the standards and processes used to implement MDC Discovery in a way that fulfills the ZeroConf requirements.

8.1 Internet Protocol Version 4 (IPv4) Address Allocation

MDC Discovery compliant devices shall be capable of receiving Internet Protocol version 4 (IPv4) addresses through static configuration on the device, Dynamic Host Configuration Protocol (DHCP) [RFC 2131], and IPv4 Link-Local Addressing (IPv4LL) [RFC 3927]. Due to IETF standardization this functionality is already available in most modern operating systems, Internet Protocol networking stacks, and Internet Protocol network hardware.

8.2 Internet Protocol Version 6 (IPv6) Address Allocation

MDC Discovery compliant devices may support Internet Protocol version 6 (IPv6). When IPv6 is supported, MDC Discovery compliant devices shall be capable of receiving IPv6 addresses through static configuration on the device, Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [RFC 3315], and IPv6 Stateless Address Autoconfiguration (SLAAC) ([RFC 4861] and [RFC 4862]). Due to IETF standardization this functionality is already available in most modern operating systems, Internet Protocol networking stacks, and Internet Protocol hardware.

8.3 Domain Name Servers

MDC Discovery utilizes name resolution services to translate human readable names to Internet Protocol addresses, translate Internet Protocol addresses into hostnames, and to discover services and Capability Interface endpoints available on the network. MDC Discovery compliant devices shall support the Domain Name System (DNS) ([RFC 1034] and [RFC 1035]) and Multicast DNS (mDNS) [RFC 6762], acting as a DNS resolver and an mDNS querier. Both DNS and mDNS have become well known and widely accepted technologies with numerous implementations and resources available to help simplify the implementation of these services within MDC Discovery compliant devices.

8.3.1 Internet Protocol Version 4 (IPv4) Name Server Determination

MDC Discovery compliant devices shall be capable of receiving DNS name server configuration through static configuration on the device and through the DHCP [RFC 2131] option 6.

8.3.2 Internet Protocol Version 6 (IPv6) Name Server Determination

MDC Discovery compliant devices may support Internet Protocol version 6 (IPv6). When IPv6 is supported, MDC Discovery compliant devices shall be capable of receiving DNS name server configuration through static configuration on the device, DNS Configuration options for DHCPv6 [RFC 3646], and IPv6 Router Advertisement Options for DNS Configuration [RFC 6106].

8.3.3 Issuing Name Resolution Requests

MDC Discovery compliant devices shall be capable of delegating the name resolution requests to each name server or load-balanced set of name servers associated to the DNS-SD browse domain. For example, if 3 name servers are associated to a particular DNS-SD browse domain and 2 of those name servers have the same `Priority`, forming a load-balanced set of name servers. Each name resolution request shall be delegated once to the load-balanced set, where the appropriate name server is determined by the SRV RR `Weight` value, as defined in RFC 2782, and once to the separate, stand-alone name server.

8.4 Determining Browse Domains

As discussed earlier, MDC Discovery is based on DNS-Based Service Discovery (DNS-SD) [RFC 6763]. Therefore, when a name resolution client wishes to search (aka. browse) for services on the network, it shall utilize the browse domains acquired by the processes described within this section as the domain name for DNS and mDNS requests.

8.4.1 Default Browse Domains

The following figure and consequent outline depict the default browse domains and methods used to acquire the default browse domains. The default browse domains shall be used by MDC Discovery compliant devices to find additional, configured, browse domains using the methods prescribed in Section 8.4.2, Finding Additional Browse Domains. The order by which the default domain names are acquired and utilized is irrelevant, provided they are all used to find the browse domains configured for the network.

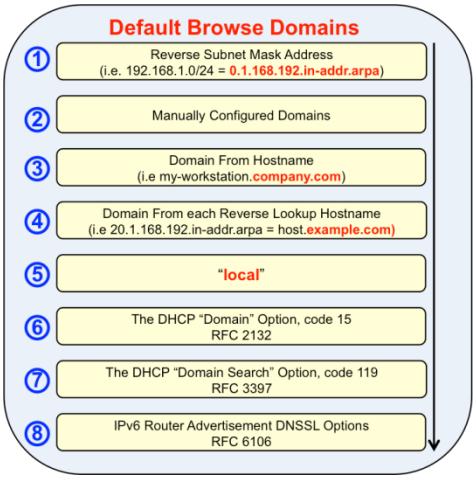

Figure 3 – Default Browse Domains

1. The Reverse Subnet Mask domain for the IPv4 and IPv6 addresses, as defined by Section 11 of the DNS-SD specification; i.e., 192.168.1.0/24 shall translate to the domain 0.1.168.192.in-addr.arpa.
2. Manually configured browse domains.
3. The domain name portion of the hostname configured for the device.
4. The domain name of each hostname returned by the reverse lookup of each IP address. ([RFC 2317], [RFC 3596 Section 2], [RFC 1033], and [RFC 1912]).
5. The mDNS Link-Local domain name "local".
6. For Internet Protocol v4 networks with DHCP services available.
    a. Add the domain name as provided by the networks DHCP services, as defined RFC 2132.
    b. Add domains found using the DHCP Domain Search Option as defined by RFC 3397.
7. For Internet Protocol v6 networks.
    a. Add domains using the IPv6 Router Advertisement Options for DNS Configuration, as defined in RFC 6106.

8.4.2 Finding Additional Browse Domains

Section 11 of the DNS-SD specification [RFC 6763] defines 3 special domain names for the purposes of browse domain discovery. These domain names are

- b._dns-sd._udp.<domain>
- db._dns-sd._udp.<domain>
- lb._dns-sd._udp.<domain>

These RR names are defined as PTR RRs within the DNS system and specify the domain names of additional browse domains for which the DNS-SD clients will use when searching for services. The MDC Discovery compliant device uses the special domain names to learn:

- The list of recommended browse domains
- A single, default browse domain
- A browse domain that should be used automatically and not presented to a user

MDC Discovery compliant devices shall use these special domain names in the manner defined in Section 11 of the DNS-SD specification [RFC 6763], shall use all of the browse domains discovered by these special domain names and may use the default browse domain names specified in Section 8.4.1, Default Browse Domains, to search for device services within the network. For the best performance, MDC Discovery devices should search the browse domains in the following order.

1. Browse domains manually configured for the MDC Discovery client or device
2. The browse domains specified by the db._dns-sd._udp.<domain> special domain names
3. The browse domains specified by the lb._dns-sd._udp.<domain> special domain names
4. The browse domains specified by the b._dns-sd._udp.<domain> special domain names
5. The mDNS Link-Local domain name "local"
6. The default browse domains

8.4.3 Finding Additional Name Servers

MDC Discovery devices shall use DNS-based Service Discovery (DNS-SD) [RFC 6763] to locate additional DNS name servers within each and every browse domain. To find the additional DNS name servers the MDC Discovery device shall search for services of type `_domain._udp` within each browse domain. Each uniquely named service instance will represent a distinct DNS name server. Each DNS name server found should be used as the name server for name resolution requests targeted for one of the browse domains indicated by

the special domain names defined in Section 8.4.2, Finding Additional Browse Domains, which were found within the browse domain in which the DNS name server was found and for name resolution requests specifying any of the default browse domains other than “local”. For example, if the default browse domain “mdc.spmte.org” were found by querying the `db._dns-sd._udp.local`, `b._dns-sd._udp.local`, or `lb._dns-sd._udp.local` PTR RRs, then all DNS name servers found by querying the service type name `_domain._udp.local` would be utilized for name resolution requests targeted for the “mdc.smpte.org” domain.

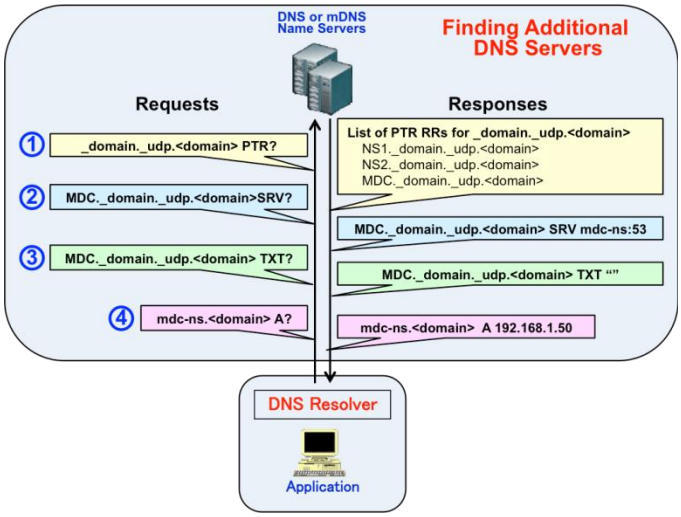

Figure 4 – Finding Additional Name Servers

As illustrated in Figure 4, a MDC Discovery device issues a DNS query for PTR RRs with the name `_domain._udp.<browse domain>`. The DNS name server returns the list of PTR RRs and finally the MDC Discovery device queries the DNS name server for the SRV and TXT RR pairs representing the service instances identified by the PTR RRs.

8.4.3.1 Finding Additional Name Servers Using Just the SRV Resource Record

MDC Discovery devices may also use the methods described in RFC 2782 to find additional DNS name servers. As described in RFC 2782, the client issues DNS queries for SRV RRs with the name `_domain._udp.<browse domain>` and as defined in Section 8.4.3, Finding Additional Name Servers, the DNS name servers found shall be used for queries targeted to the browse domains indicated by the special domain names specified in Section 8.4.2, Finding Additional Browse Domains, for the browse domain in which the DNS name server was found and for name resolution requests specifying any of the default browse domains other than “local”. Please note that because this is not DNS-SD, the SRV RRs returned from these queries should not have corresponding TXT RRs.

8.5 Finding Devices (Browsing)

As stated above MDC Discovery is based on DNS-Based Service Discovery (DNS-SD) [RFC 6763]. Therefore, the procedure used to find MDC Discovery services shall be the same process used for browsing DNS-SD services.

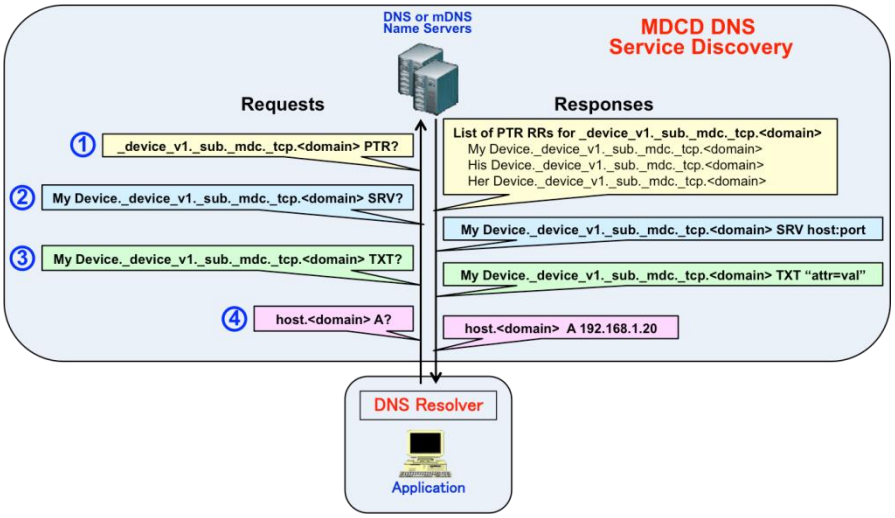

Figure 5 – MDC Device Discovery

As illustrated in Figure 5, when a resolver wishes to browse for MDC Devices or gain access to a Device Capability Interface endpoint of a device, it shall issue a DNS or mDNS name resolution request for the PTR RRs of the name `_device_v1._sub._mdc._tcp.<domain>`. In this example, the DNS or mDNS name server(s) respond with three PTR RRs containing the service instance names:

- `My\ Device_device_v1._sub._mdc._tcp.<domain>`
- `His\ Device_device_v1._sub._mdc._tcp.<domain>`
- `Her\ Device_device_v1._sub._mdc._tcp.<domain>`

The MDC Discovery resolver will manually or automatically select one or more of the results and shall issue name resolution requests for the SRV and TXT RRs with a domain name equal to the PTR RRs `Instance.Service` field. The SRV RR provides the client with the hostname and IP port on which the Device Capability Interface endpoint resides and the TXT RR provides the additional information required to access the Capability Interface endpoint, such as the wire level protocol and the URL path.

8.5.1 Finding Devices (Browsing) Example (Informative)

The following example illustrates how the “nslookup” command line utility can be utilized to demonstrate MDC Discovery and DNS-SD browsing. The “nslookup” utility is a widely known administration utility available on almost all operating systems.

```
% nslookup -type=ptr _device_v1._sub._mdc._tcp.example.org
Server:  my.nameserver
Address:  192.168.0.1

_device_v1._sub._mdc._tcp.example.org    Name = My Device._device_v1._sub._mdc._tcp.example.org
_device_v1._sub._mdc._tcp.example.org    Name = His Device._device_v1._sub._mdc._tcp.example.org
_device_v1._sub._mdc._tcp.example.org    Name = Her Device._device_v1._sub._mdc._tcp.example.org

% nslookup -type=srv My\ Device._device_v1._sub._mdc._tcp.example.org
Server:  my.nameserver
Address:  192.168.0.1

My Device._device_v1._sub._mdc._tcp.example.org    SRV service location:
priority       = 0
weight         = 1
port           = 8080
srv hostname   = my-device.example.org  internet address = 192.168.0.61

% nslookup -type=txt My\ Device._device_v1._sub._mdc._tcp.example.org
Server:  my.nameserver
Address:  192.168.0.1

My Device._device_v1._sub._mdc._tcp.example.org  text = "txtvers=1" "rn=urn:smpte:urn:my-device"
                                                       "proto=mdcp" "path=/Device_v1"
```

This example illustrates the same example as the diagram in Figure 5, which can be used to browse or discover any MDC Capability Interface registered with the DNS or mDNS name servers.

8.6 Making Devices Discoverable

DNS-Based Service Discovery and consequently MDC Discovery work by allowing clients to query the naming infrastructure for services that implement specific interfaces. However, in order for services to be discoverable, they must first be made available to the naming infrastructure. The method by which services are made available to the naming infrastructure depends on whether the naming infrastructure is based on DNS or mDNS.

8.6.1 DNS

DNS is a centralized system, where the resolver or name resolution client issues requests to one or more naming servers. Services may be statically configured or dynamically registered with the naming server(s).

8.6.1.1 DNS UPDATE [RFC 2136]

DNS UPDATE [RFC 2136] is the means by which the DNS naming server(s) may be dynamically configured. MDC Discovery compliant devices that utilize DNS UPDATE as a means of configuring the DNS infrastructure shall issue DNS UPDATE requests in the order:

1. Add or update the A or AAAA RRs, assigning the hostname of the device to the appropriate IP addresses.
2. Add or update the MDC service TXT RR using the service's human readable name and the service type name as the `Instance.Service` name, e.g. My Device._device_v1._sub._mdc._tcp, setting the TXT RR attributes as defined by Section 7.3, Text (TXT) Resource Record.
3. Add or update the MDC service SRV RR using the service's human readable name and the service type name as the `Instance.Service` name, e.g. My Device._device_v1._sub._mdc._tcp, and setting the SRV RR fields as defined by Section 7.2, Service (SRV) Resource Record.
4. Add or update the MDC service type PTR RR using "_mdc._tcp" as the value of the `Service` field and pointing to the `Instance.Service` name. The `Instance.Service` value shall conform to the specification in Section7.1, Pointer (PTR) Resource Record.
5. Add or update the MDC Capability Interface's service subtype PTR RR using the Capability Interface service subtype name, as defined in Section 6.1, Capability Interfaces and Service Subtypes, as the value of the `Service`field and pointing to the `Instance.Service` name. The `Instance.Service` value shall conform to the specification in Section 7.1, Pointer (PTR) Resource Record.

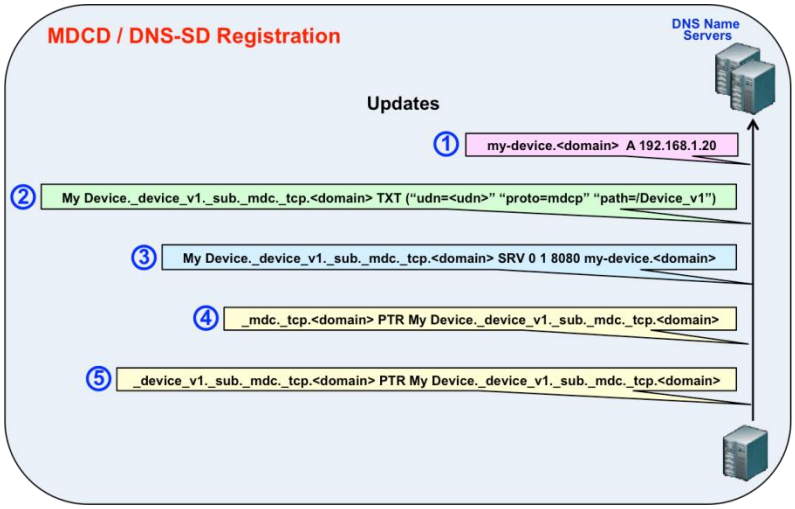

Figure 6 – MDC Discovery Device using DNS UPDATE

As shown in Figure 6 and presented in the list above, the MDC Discovery DNS RRs are added to the DNS infrastructure in the reverse order by which they are requested during the discovery / browse process. This order is significant as it helps to ensure that partially configured services are not exposed.

8.6.1.1.1 Security Considerations

Without security extensions DNS UPDATE is inherently insecure. MDC Discovery compliant devices or the DNS name servers utilized by them may implement Secure Domain Name System Dynamic Update [RFC 3007] to secure the DNS UPDATE operations. It is highly recommended that if the DNS name server or the MDC Discovery compliant devices are located within an unprotected network, the DNS name server and MDC Discovery devices should utilizes Secure Domain Name System Dynamic Update [RFC 3007] for DNS UPDATE operations.

8.6.1.2 Static Configuration

All DNS name servers provide some means of configuration. This configuration may be used to statically configure the DNS server for the MDC Devices to be discovered. To manually configure a the DNS name

server for MDC Discovery, the DNS administrator shall configure the same DNS RRs as are defined in Section 8.6.1.1, DNS UPDATE [RFC 2126].

1. A or AAAA RRs shall be added, assigning the hostname of the device to the appropriate IP addresses.
2. A TXT RR shall be added, using the service's human readable name and the service type name as the `Instance.Service` name, e.g. My Device._device_v1._sub._mdc._tcp and setting the TXT RR attributes as defined by Section 7.3, Text (TXT) Resource Record.
3. A SRV RR shall be added, using the service's human readable name and the service type name as the `Instance.Service` name, e.g. My Device._device_v1._sub._mdc._tcp and setting the SRV RR fields as defined by Section 7.2, Service (SRV) Resource Record.
4. A PTR RR pointing to the `Instance.Service` name shall be added, specifying "_mdc._tcp" as the `Service` field. The `Instance.Service` value shall conform to the specification in Section 7.1, Pointer (PTR) Resource Record.
5. A PTR RR pointing to the `Instance.Service` name shall be added, specifying the service subtype name of the Capability Interface as defined in Section 6.1, Capability Interfaces and Service Subtypes, as the value of the `Service`field. The `Instance.Service` value shall conform to the specification in Section 7.1, Pointer (PTR) Resource Record.

8.6.1.2.1 "BIND" Configuration Example

The following example shows the DNS RRs needed to configure the DNS name server for MDC Discovery using the configuration format for the "BIND" software. "BIND" is the most well-known and widely adopted DNS name server implementation available on the Internet.

```
device1                                 IN A 192.168.0.20
_mdc._tcp                               IN PTR Instance._device_v1._sub._mdc._tcp
_device_v1._sub._mdc._tcp               IN PTR Instance._device_v1._sub._mdc._tcp
Instance._device_v1._sub._mdc._tcp      IN SRV 10 10 8080 device1
    TXT ("txtvers=1" "rn=urn:smpte:udn:namespace1:device1" "proto=mdcp" "path=/Device")
```

8.6.1.3 Domain Name System Security Extensions (DNSSEC)

Domain Name System Security Extensions (DNSSEC) ([RFC 4033], [RFC 4034], and [RFC 4035]) are a set of specifications that add security to specific DNS functions. MDC Discovery compliant devices or the DNS nameservers used for MDC Discovery may implement DNSSEC to verify the authoritative sources of DNS RRs, authorize DNS Resolvers, or for other miscellaneous functions. It is highly recommended that MDC Discovery implementations that expose services over the Internet, or any other insecure network, should utilize DNSSECto provide a means to verify the integrity of the DNS information.

8.6.2 Multicast DNS (mDNS)

As discussed earlier, Multicast DNS (mDNS) [RFC 6762] is a variation of DNS that utilizes Multicast UDP datagrams for name resolution. mDNS is used on the Link-Local network for decentralized name resolution. According to RFC 6763, when a MDC Discovery compliant device starts up, wakes up from sleep, or detects any change in network connectivity, the device shall broadcast the name resolution responses describing the device and the Capability Interfaces it exposes for discovery. mDNS queriers are required to cache unsolicited name resolution responses and therefore, when the MDC Discovery device broadcasts the DNS RRs describing itself, other mDNS devices on the Link-Local network cache the DNS RRs. Please refer to Section 8.6.1.1, DNS UPDATE [RFC 2126], for the detailed list of DNS RRs that shall be broadcast upon start up, waking up from sleep, or changes in network connectivity.

Annex A Bibliography (Informative)

[RFC 1033] Internet Engineering Task Force (IETF) (1987, November). IETF RFC 1033 — DNS Administrators Guide. http://tools.ietf.org/html/rfc1033.html

[RFC 1912] Internet Engineering Task Force (IETF) (1996, February).IETF RFC 1912 — Common DNS Operational and Configuration Errors. http://tools.ietf.org/html/rfc1912.html

[RFC 2317] Internet Engineering Task Force (IETF) (1998, March).IETF RFC 2317 — Classless IN-ADDR.ARPA delegation. http://tools.ietf.org/html/rfc2317.html

[RFC 3596] Internet Engineering Task Force (IETF) (2003, October). IETF RFC 3596 — DNS Extensions to Support IP version 6. http://tools.ietf.org/html/rfc3596.html

[Zeroconf WG] Internet Engineering Task Force (IETF) (2003, July). IETF Working Group — Zeroconf. http://tools.ietf.org/wg/zeroconf/

[Zeroconf] Zero Configuration Networking (Zeroconf) (2003, July). http://www.zeroconf.org

Annex B DNS Primer (Informative)

B.1 What is the Domain Name System (DNS)?

The Domain Name System (DNS) is a hierarchical, distributed, database system that constitutes the name resolution infrastructure of the Internet. DNS is designed to associate domain names with various pieces of information, but its primary purpose is to associate network based resources with an easy to remember, human-readable, domain name, known as a hostname, and to translate those hostnames into the network resource's numerical network address or IP address. For example, in order for the SMPTE web site (www.smpte.org) to be accessed by a web browser, such as Internet Explorer, Chrome, or Firefox, the hostname "www.smpte.org" must be translated into the IP address that is associated to the "www" hostname within the "smpte.org" domain (107.21.249.46).

Before DNS was invented, hostname to IP address resolution was performed using a *hosts file* located on each computer. The *hosts file*, named HOSTS.TXT was managed by the SRI (now SRI International) NIC (Network Information Centre) and distributed to every computer on the network. By 1982, the management of a centralized, hand crafted, *hosts file* became unsustainable and a more scalable system for name resolution was required. In 1983, Paul Mockapetris invented DNS, at the request of Jon Postel, to address the issues pertaining to *hosts files*. Modern computer systems still contain *hosts files*, however they are primarily used to store *localhost* name resolution information, but may be used to store local network addresses. For the purposes of name resolution, the *hosts file* takes precedence over DNS, meaning that if a hostname entry exists in the *hosts file*, the computer will not delegate the name query to DNS, but will instead use the value contained within the *hosts file*.

DNS is comprised of two sub-systems; the name server, which manages the domain hierarchy and name resolution and the resolver, which acts as the client for the name server, issuing name resolution requests. It is the name server that implements the distributed, hierarchical database structure. Figure B.1 demonstrates how the domain name "www.smpte.org" is resolved to the IP address 107.21.249.46.

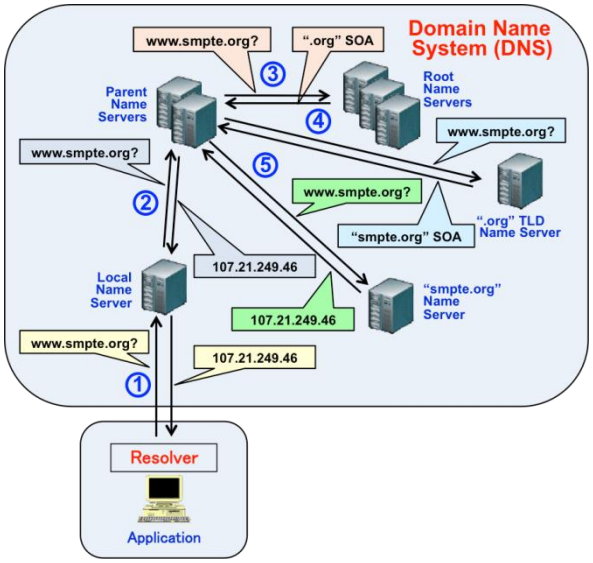

Figure B.1 – DNS Name Resolution for "www.smpte.org"

As shown in this figure, when an application desires to obtain an IP address for a hostname, "www.smpte.org" in this example, the application's resolver issues a name resolution request to the pre-configured Local Name Server (DNS Name Server). If the Local Name Server already has response for the name resolution request

cached within memory, it will immediately return the response. However, if the Local Name Server does not have the response for the name resolution request cached, it will delegate the name resolution request to its pre-configured Parent Name Server. If the Parent Name Server does not have the response cached, it will in turn delegate the name resolution request to its pre-configured Parent Name Server. The name resolution request will continue to be delegated to Parent Name Servers until either a response is known or the Root Name Servers, defined for the Internet, are reached. The Root Name Servers are DNS name server assigned by the Internet Assigned Numbers Authority (IANA) to manage the Top Level Domains (TLDs), such as ".com", ".org", and ".net" Since the Top Level Domain (TLD) for the SMPTE web site's hostname ("www.smpte.org") is ".org", the Root Name Server assigned as the Start of Authority (SOA) for the ".org" Top Level Domain (TLD) will respond with the Start of Authority Resource Record (SOA RR) (see Section B.2.1, Start of Authority (SOA) Resource Record), for the "smpte.org" domain. The Local Name Server then uses the SOA RR to determine the DNS name servers assigned to manage the "smpte.org" domain and will ask those name servers for the IP addresses associated to the hostname "www". The "smpte.org" name servers then return "107.21.249.46" to the Local Name Server and the final response is returned to the resolver that initiated the name resolution request. The Local Name Server will cache the response for the "www.smpte.org" hostname for the Time To Live (TTL) number of seconds defined in the "www.smpte.org" Address Resource Record (A RR) or for the default Time To Live (TTL) value specified in the "smpte.org" SOA RR. If a resolver later attempts to lookup "www.smpte.org", within the TTL period, the Local Name Server will immediately return the response cached by the initial name resolution request.

It is important to reiterate that while DNS is usually used for name resolution, it can also be used as a general, distributed, hierarchical database, storing almost any kind of data, for nearly any purpose.

B.2 DNS Resource Records

This section discusses some of the basic DNS Resource Records (DNS RRs). DNS Resource Records (DNS RRs) are the basic data elements stored within a DNS name server. Each DNS Resource Record (DNS RR) represents a single data point of a certain format and has a distinct type associated to it that identifies the records format, purpose, and information contained therein.

This section is not intended to be a comprehensive list of DNS RRs, but instead it is intended to give the reader a basic understanding of DNS RRs, how they are configured and how they are used within the confines of the Domain Name System. For further details on all of the available DNS Resource Record (DNS RR) types, please refer to the associated IETF RFCs or documentation for a relevant DNS name server.

The format of the DNS Resource Record (DNS RR) is represented according to the Resource Record configuration format used by the "BIND" software, the most well-known and widely adopted DNS name server implementation and the name resolution request/response examples are created using the "nslookup" utility, the standard network command-line resolver, available on almost all computer operating systems.

B.2.1 Start of Authority (SOA) Resource Record

The Start of Authority (SOA) Resource Record is used to specify the authoritative information about a DNS Zone, aka Domain. The SOA RR includes the primary Name Server, serial number, and several timers, such as the default Time To Live (TTL) for RRs associated to the domain.

B.2.1.1 Start of Authority Resource Record Format

The format of the SOA Resource Record ([RFC 1034] and [RFC 1035]) shall be in the form:

```
[Domain] [Class] "SOA" MName [RName] ( Serial Refresh Retry Expiry TTL )
```

`Domain`	Specifies the domain name of the name server that is the primary source for the zone. (Optional, may be blank or '@').
`Class`	Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to "IN", without the quotes, if unspecified. (Optional)
`"SOA"`	Indicates that this RR is a Start of Authority RR. The value shall be equal to "SOA", without the quotes.
`MName`	Specifies the domain name of the primary name server for the zone/domain.
`RName`	Specifies the domain name of the mailbox for the person administering the zone. (Optional)
`Serial`	Specifies the 32-bit unsigned integer, ranging from 0 to 4294967295 (2^{32} - 1), version number of the original copy of the zone.
`Refresh`	Specifies the 32-bit unsigned integer, ranging from 0 to 4294967295 (2^{32} - 1), which is the interval in which the zone/domain should be refreshed.
`Retry`	Specifies the 32-bit unsigned integer, ranging from 0 to 4294967295 (2^{32} - 1), which is the amount of time to wait before retrying a failed zone/domain refresh.
`Expiry`	Specifies the 32-bit unsigned integer, ranging from 0 to 4294967295 (2^{32} - 1), which is the maximum amount of time in which the data for this zone/domain is valid, the maximum TTL.
`TTL`	Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the minimum number of seconds that a RR for this zone/domain can be cached for before the information should be re-queried. Note that a value of '0' indicates that the RR should not be cached and shall be discarded immediately.

B.2.1.1.1 Start of Authority Resource Record Example

Suppose a name server `my.nameserver` is configured to be the SOA for the domain `example.org`.

```
example.com.       IN      SOA     ns.example.com.hostmaster.example.com
    (
        2013041050 ; serial number
        172800     ; refresh = 2 Days
        900        ; update retry = 15 Minutes
        1209600    ; expiry = 2 Weeks
        3600       ; nxdomain TTL = 1 Hour
    )
```

Where text following the semi-colon ';' is a comment statement and is ignored by the DNS name server.

A query issued by the resolver (`nslookup`) would have the following response from the name server.

```
% nslookup -type=soa example.org
Server:  my.nameserver
Address:  192.168.0.1

example.org
   origin = ns.example.org
   mail addr = hostmaster.example.org
   serial = 2013041050
   refresh = 172800
   retry = 900
   expire = 1209600
   minimum = 3600
```

Where `nslookup` were executed with the argument “example.org” and a type option specifying a query for the SOA RR type. The response from the name server shows that the origin/name server is `ns.example.org`, the minimum/default TTL is 3600 seconds (1 hour) and zone/domain data is valid for `1209600` seconds (2 weeks).

B.2.2 Address (A) Resource Record

The Address (A) Resource Record is used to map a hostname to its 32-bit IPv4 addresses. Address RRs may be used to map a hostname to one or more IP addresses, or one or more IP addresses to a single hostname. Multiple IP addresses are typically mapped to a single hostname for load balancing and the DNS name server will return the full list of IP addresses for the hostname when the hostname is queried. The assignment of multiple hostnames to a single IP address is typically used for virtual hosting or to expose multiple services on a single server, using meaningful names for each service, such as www, smtp, and dns.

B.2.2.1 Address Resource Record Format

The format of the A Resource Record ([RFC 1034] and [RFC 1035]) shall be in the form

```
[Owner] [TTL] [Class] "A" Address
```

`Owner` Specifies the hostname of the Resource Record (RR). (Optional)

`TTL` Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the number of seconds that this RR can be cached before the information should be requeried from the source. Note that a value of ‘0’ indicates that the RR should not be cached and shall be discarded immediately. (Optional)

`Class` Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to “IN”, without the quotes, if unspecified. (Optional)

`"A"` Indicates that this RR is an Address RR. The value shall be equal to “A”, without the quotes.

`Address` Specifies the 32-bit IPv4 address.

B.2.2.1.1 Address Resource Record Example

Suppose a name server `my.nameserver` is configured to be the SOA for the domain `example.org`.

```
host        IN      A       66.179.20.124
```

A query issued by the resolver (`nslookup`) would have the following response from the name server.

```
% nslookup -type=a host.example.org
Server:  my.nameserver
Address:  192.168.0.1

Name:     host.example.org
Address:  66.179.20.124
```

Where `nslookup` were executed with the argument “host.example.org” and a type option specifying a query for the A RR type. The response from the name server shows that the IP address for `host.example.org` is equal to `66.179.20.124`.

B.2.3 IPv6 Address (AAAA) Resource Record

The IPv6 Address (AAAA) Resource Record is used to map a hostname to its 128-bit IPv6 addresses. IPv6 address RRs may be used to map a hostname to one or more IPv6 addresses, or one or more IPv6 addresses to a single hostname. Multiple IPv6 addresses are typically mapped to a single hostname for load balancing and the DNS name server will return the full list of IPv6 addresses for the hostname when the hostname is queried. The assignment of multiple hostnames to a single IPv6 address is typically used for virtual hosting or to expose multiple services on a single server, using meaningful names for each service, such as www, smtp, and dns.

B.2.3.1 IPv6 Address Resource Record Format

The format of the AAAA Resource Record [RFC 3596] shall be in the form:

```
[Owner] [TTL] [Class] "AAAA" Address
```

`Owner` Specifies the hostname of the Resource Record (RR). (Optional)

`TTL` Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the number of seconds that this RR can be cached before the information should be requeried from the source. Note that a value of '0' indicates that the RR should not be cached and shall be discarded immediately. (Optional)

`Class` Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to "IN", without the quotes, if unspecified. (Optional)

`"AAAA"` Indicates that this RR is an IPv6 Address RR. The value shall be equal to "AAAA", without the quotes.

`Address` Specifies the 128-bit IPv6 address.

B.2.3.1.1 IPv6 Address Resource Record Example

Suppose a name server `my.nameserver` is configured to be the SOA for the domain `example.org`.

```
host        IN      AAAA    4321:0:1:2:3:4:567:89ab
```

A query issued by the resolver (`nslookup`) would have the following response from the name server.

```
% nslookup -type=aaaa host.example.org
Server:  my.nameserver
Address:  192.168.0.1

Name:     host.example.org
Address:  4321:0:1:2:3:4:567:89ab
```

Where `nslookup` were executed with the argument "host.example.org" and a type option specifying a query for the AAAA RR type. The response from the name server shows that the IP address for `host.example.org` is equal to `4321:0:1:2:3:4:567:89ab`.

B.2.4 Canonical Name (CNAME) Resource Record

Canonical Name (CNAME) Resource Records are used to specify aliases for other domain names or hostnames. Here the term "canonical" is used to mean a more generally accepted or standard name.

Note that CNAME RRs that point to other CNAME RRs are permitted, though they are strongly discouraged, to prevent unresolvable loops that may be caused by mutual references.

B.2.4.1 Canonical Name Resource Record Format

The format of the CNAME Resource Record ([RFC 1034] and [RFC 1035]) shall be in the form:

```
[Owner] [TTL] [Class] "CNAME" Name
```

`Owner` Specifies the hostname of the Resource Record (RR). (Optional)

`TTL` Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the number of seconds that this RR can be cached before the information should be requeried from the source. Note that a value of '0' indicates that the RR should not be cached and shall be discarded immediately. (Optional)

`Class` Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to "IN", without the quotes, if unspecified. (Optional)

`"CNAME"` Indicates that this RR is a Canonical Name RR. The value shall be equal to "CNAME", without the quotes.

`Name` Specifies the canonical domain name.

B.2.4.1.1 Canonical Name Resource Record Example

Suppose a name server `my.nameserver` is configured to be the SOA for the domain `example.org`.

```
www IN      CNAME   @
@   IN      A       107.21.249.46
```

Where the symbol '@' is a shorthand for the SOA domain name.

A query issued by the resolver (`nslookup`) would have the following response from the name server.

```
% nslookup -type=cname www.example.org
Server:  my.nameserver
Address:  192.168.0.1

www.example.org   canonical name = example.org

% nslookup -type=a www.example.org
Server:  my.nameserver
Address:  192.168.0.1

Name:     www.example.org
Address:  107.21.249.46
Aliases:  example.org

% nslookup -type=a example.org
Server:  my.nameserver
Address:  192.168.0.1

Name:     example.org
Address:  107.21.249.46
```

Where `nslookup` were executed with the argument "www.example.org" and a type option specifying a query for the CNAME RR type. The response from the name server shows that the canonical domain name of `www.example.org` is `example.org`.

B.2.5 Pointer (PTR) Resource Record

Pointer (PTR) Resource Records are used to map one domain name to another domain name. In this aspect, the PTR RR is similar to the CNAME RR. But unlike the CNAME RR, the DNS Name Server does not conduct further processing for the PTR RR, just returning the alias domain name.

B.2.5.1 Pointer Resource Record Format

The format of the PTR Resource Record ([RFC 1034] and [RFC 1035]) shall be in the form:

```
[Owner] [TTL] [Class] "PTR" Name
```

Owner	Specifies the hostname of the Resource Record (RR). (Optional)
TTL	Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the number of seconds that this RR can be cached before the information should be requeried from the source. Note that a value of '0' indicates that the RR should not be cached and shall be discarded immediately. (Optional)
Class	Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to "IN", without the quotes, if unspecified. (Optional)
"PTR"	Indicates that this RR is a Pointer RR. The value shall be equal to "PTR", without the quotes.
Name	Specifies the domain name.

B.2.5.1.1 Pointer Resource Record Example

The most common use of the PTR RR is for the implementation of reverse DNS lookups, where a specified IP address is resolved to its assigned (canonical) hostname. Suppose a hostname `example.org` has an IP address of `107.21.249.46`. In order for the reverse DNS lookup to be conducted a specialized domain called `in-addr.arpa` is provided. Because an IP address is refined from left to right; i.e., the leftmost byte is most significant, which is an opposite order of the domain name, where the rightmost is the top-level domain, a query issued by the resolver to the Local Name Server is represented as `46.249.21.107.in-addr.arpa`, with the IP address octets presented in reverse order.

This "domain name" is also resolved in a recursive fashion as is demonstrated in Figure B.1 – DNS Name Resolution for "www.smpte.org**"**; i.e., the Local Name Server will delegate unknown requests to its Parent Name Server. When the Local Name Server finally reaches the `249.21.107.in-addr.arpa` Name Server, it responds to the Local Name Server with `example.org`.

Suppose a name server `my.nameserver` is configured to be the SOA for the reverse map domain `249.21.107.in-addr.arpa.`

```
46  PTR     www.example.org.
```

Where "46" is the last octet of the host's IP address and consequently treated as a hostname within the reverse map domain.

A query issued by the resolver (`nslookup`) would have the following response from the name server.

```
% nslookup -type=ptr 46.249.21.107.in-addr.arpa my.nameserver
Server:  pdns1.ultradns.net
Address:  204.74.108.1

46.249.21.107.in-addr.arpa      name = www.example.org
```

Where `nslookup` were executed with the argument "46.249.21.107.in-addr.arpa" and a type option specifying a query for the PTR RR type. The response from the name server shows that the domain name of `46.249.21.107.in-addr.arpa` is `www.example.org`.

It should be noted that because the function of the PTR RR is just a mapping of one domain name to another, it is also used for a simple query of another domain name. Suppose a name server `my.nameserver` is configured to be the SOA for the domain `example.org`.

```
_http._tcp  IN      PTR     A\ web\ page._http._tcp
```

A query issued by the resolver (nslookup) would have the following response from the name server:

```
% nslookup -type=ptr _http.tcp.example.org
Server:  my.nameserver
Address:  192.168.0.1

_http._tcp.example.org.      name = A web page._http._tcp.example.org.
```

Where nslookup were executed with the argument "_http._tcp.example.com" and a type option specifying a query for the PTR RR type. The response from the name server shows that another domain name of _http._tcp.example.com is A web page._http._tcp.example.org.

B.2.6 Service (SRV) Resource Record

Service (SRV) Resource Records are used to define the location, i.e. the hostname and the port number, of a service. The SRV RR can be used to represent any service that runs on a specific port, but by itself does not support multiple services running on the same port. In order to support multiple services that run on the same port, an additional mechanism is required, which is further discussed in Annex C DNS-SD Printer (Informative).

B.2.6.1 Service Resource Record Format

The format of the SRV resource record, as defined by RFC 2782, shall be in the form:

Instance._Service._Proto TTL Class SRV Priority Weight Port Target

Service	The symbolic name of the service type. Shall conform to the naming convention established for service subtype names specified in Section 6.1, Capability Interfaces and Service Subtypes.
Proto	Specifies the transport protocol of the desired service, which is usually either TCP or UDP.
TTL	Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the number of seconds that this RR can be cached before the information should be re-queried from the source. Note that a value of '0' indicates that the RR should not be cached and shall be discarded immediately. (Optional)
Class	Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to "IN", without the quotes, if unspecified.
"SRV"	The SRV record type specification. The value shall be equal to "SRV", without the quotes.
Priority	Shall contain the Priority of this Resource Record relative to other SRV Resource Records sharing the same Service. Zero (0) is the highest priority.
Weight	Shall specify the weight of the SRV record relative to other SRV Resource Records sharing the same Service and Priority.
Port	Shall contain the value of the Internet Protocol Port used by the endpoint on the Target host.
Target	Shall contain the domain name of target host.

B.2.6.1.1 Service Resource Record Example

Suppose a name server my.nameserver is configured to be the SOA for the domain example.org.

```
_http._tcp  IN      SRV     1       92      80      www
www         IN      A       107.21.249.46
```

A query issued by the resolver (nslookup) would have the following response from the name server.

```
% nslookup -type=srv _http._tcp.example.org
Server:  my.nameserver
Address:  192.168.0.1

_http._tcp.example.org    SRV service location:
priority       = 1
weight         = 92
port           = 80
srv hostname   = www.example.org  internet address = 107.21.249.46
```

Where nslookup were executed with the argument "_http._tcp.example.com" and a type option specifying a query for the SRV RR type. The response from the name server shows that a HTTP service is exposed on the host www.example.com (107.21.249.46) on port 80.

SRV RRs alone can also be used for any service that is exposed via one port and may also be used for the load balancing of multiple services. For example, if the name server is configures with the SRV RRs of

```
_http._tcp      IN  SRV  0  2  80    www.example.org.
                IN  SRV  0  1  80    www2.example.org.
                IN  SRV  1  1  8000  www3.example.org.
```

This indicates that www.example.org or www2.example.org should be used first and that www.example.org should be used 2 times as often as www2.example.org. This is indicated through the priority of 0 and the weight of 2 and 1, respectively; i.e., the priority of www.example.org and www2.example.org are the same and the weight of www.example.org is 2 times larger than the weight of www2.example.org. If both www.example.org and www2.example.org are unavailable, then www3.example.org should be used. Please note that www3.example.org utilizes a different port (8000) than www.example.org and www2.example.org. Please also note that if the name value before the Class (IN) is omitted, it is assumed to be inherited from the DNS RR that precedes it; i.e., that exists immediately above it, in the configuration file.

B.2.7 Text (TXT) Resource Record

Text (TXT) Resource Records are used to associate textual data to a domain name.

B.2.7.1 Text (TXT) Resource Record Format

The format of the TXT Resource Record [RFC 1035] shall be in the form

```
[Owner] [TTL] [Class] "TXT" ( Text Text ... )
```

Owner	Specifies the hostname of the Resource Record (RR). (Optional)
TTL	Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the number of seconds that this RR can be cached before the information should be requeried from the source. Note that a value of '0' indicates that the RR should not be cached and shall be discarded immediately. (Optional)
Class	Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to "IN", without the quotes, if unspecified. (Optional)
"TXT"	The TXT record type specification. The value shall be equal to "TXT", without the quotes.
Text	Shall contain textual data, as prescribed by RFC 1035.

B.2.7.1.1 Text Resource Record Example

Suppose a name server `my.nameserver` is configured to be the SOA for the domain `example.org`.

```
host        IN      TXT     ( "Some" "Text" )
```

A query issued by the resolver (`nslookup`) would have the following response from the name server.

```
% nslookup -type=txt host.example.org.
Server:  my.nameserver
Address:  192.168.0.1

host.example.org  text = "Some" "Text"
```

Where `nslookup` were executed with the argument "host.example.com" and a type option specifying a query for the TXT RR type. The response from the Name Server shows that two distinct text strings `"Some"` and `"Text"` are associated with the hostname `host.example.org`.

B.2.8 Next-Secure (NSEC) Resource Record

Next-Secure (NSEC) Resource Records [RFC 4034] should be used to indicate authoritative Resource Records present for a RR. NSEC RRs contain two separate values, the Next Domain Name, in the canonical ordering of the zone, and the set of RR types present for the NSEC records Next Domain Name. For example, the Next Domain Name could be the hostname associated to a PTR RR and a RRSet indicating the authoritative RR types that are available for the Next Domain Name, such as SRV, TXT, A, AAAA, etc.

B.2.8.1 Next-Secure (NSEC) Resource Record Format

The format of the NSEC Resource Record ([RFC 1035] and [RFC 2181]) shall be in the form

```
Owner TTL Class "NSEC" NDName RRSet
```

`Owner` Specifies the hostname of the Resource Record (RR). (Optional)

`TTL` Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the number of seconds that this RR can be cached before the information should be requeried from the source. Note that a value of '0' indicates that the RR should not be cached and shall be discarded immediately. (Optional: Defaults to the Zone TTL)

`Class` Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to "IN", without the quotes, if unspecified. (Optional)

`"NSEC"` Indicates that this RR is a NSEC RR. The value shall be equal to "`NSEC`", without the quotes.

`NDName` Shall contain the Next Domain Name that has authoritative data for the RR associated to the name specified in the `Owner` field.

`RRSet` Shall contain the list of authoritative RR types associated to the Next Domain Name specified in the `NDName` field.

B.2.8.1.1 Next-Secure Resource Record Example

Suppose a name server `my.nameserver` is configured to be the SOA for the domain `example.org`.

```
_http._tcp IN      NSEC    host    A AAAA
```

A query issued by the resolver (`nslookup`) would have the following response from the name server.

```
% nslookup -type=nsec _http._tcp.example.org.
Server:  my.nameserver
Address:  192.168.0.1

_http._tcp.example.org  rdata = host.examle.org A AAAA
```

Where `nslookup` were executed with the argument "_http._tcp.example.com" and a type option specifying a query for the NSEC RR type. The response from the Name Server shows that there is one host `host.example.org` and two resource record types `A` and `AAAA`.

Annex C DNS-SD Primer (Informative)

C.1 What is the DNS-based Service Discovery (DNS-SD)?

DNS-Based Service Discovery (DNS-SD) [RFC 6763] is a means of using the Domain Name System (DNS) ([RFC 1034] and [RFC 1035]) to search the network for services. DNS-SD allows a client to search the Domain Name System (DNS) for a desired service type and the client will receive a list of specific service instances. The client can then choose from the service instances in an automated fashion or through user interaction. This process of searching for services is commonly referred to as “browsing”.

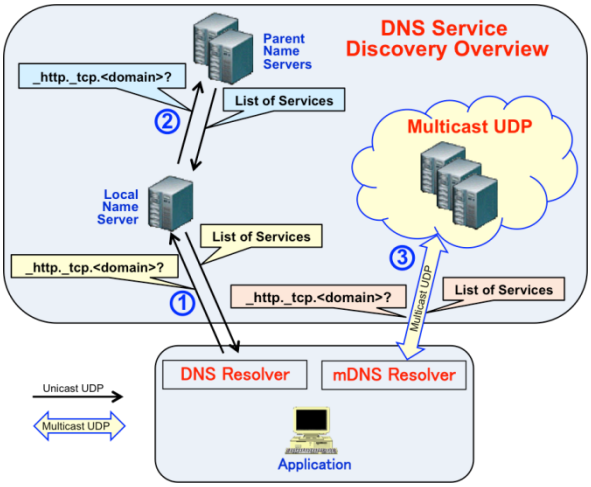

Figure C.1 – DNS-Based Service Discovery Overview

Over the years, there have been many proposed methods of network based service discovery, but none of them have achieved ubiquity in the marketplace. DNS-SD has the advantage of utilizing DNS as its base protocol and uses standard DNS Resource Record types. DNS is the most widely accepted name resolution system utilized in IP networks and is typically present on all IP networks. DNS is also well understood by Information Technology professionals and DNS name resolution clients are built into most modern operating systems.

The merits of use of DNS for the service discovery are summarized below:

- ✓ Service discovery requires a central aggregation server. DNS already has one: a DNS server.
- ✓ Service discovery requires a service registration protocol. DNS already has one: DNS Dynamic Update.
- ✓ Service discovery requires a query protocol. DNS already has one: DNS queries.
- ✓ Service discovery requires security mechanisms. DNS already has security mechanisms: DNSSEC.
- ✓ Service discovery requires a multicast mode for ad hoc networks. Using DNS-SD in conjunction with Multicast DNS (mDNS) [RFC 6762] provides this.

DNS-SD has been designed so that no changes have been made to the DNS message structure. DNS-SD also uses existing DNS Resource Record types and therefore there is no need to define new Resource Record types or operation codes.

Please Note: When using DNS-SD with DNS, the DNS-SD name server(s) do not have to be the same DNS name server(s) used to provide the organization’s primary name resolution services. DNS-SD defines 5 PTR Resource Record names that are used by the client to discover the browse and registration domains.

C.1.1 The DNS-SD Service (SRV) Resource Record

Traditional SRV Resource Records (SRV RRs) [RFC 2782] are useful for locating instances of a particular service type, but the assumption is made that all of the service instances are effectively indistinguishable from one another, providing the same exact service to the client. For example, SRV RRs with `_http._tcp.<domain>` would allow a client to discover web servers that provide the exact same set of web pages. Another assumption made by the SRV RR defined in RFC 2782 is that each service is assigned a distinct port number. It is assumed that each port exposes one and only one service endpoint, meaning that each IP port functions as a single operation protocol, such as FTP or DNS. This is problematic, as modern development methods and Service Oriented Architectures (SOA) share the same IP port for multiple service endpoints. For example, Web Services are typically exposed on the same IP port as other HTTP services, usually port 80, and all SOAP messages, regardless of the operation, are all transmitted via that IP port. The issue with traditional SRV RRs, regarding SOAP, is similar to the Web Page issue with web servers. DNS SRV RRs can be used to point to the web sever, but cannot be used to point to a specific web page on that web server. DNS-SD addresses this issue by using human-readable service instance names to identify distinct service instances and TXT Resource Records (TXT RRs) to contain instance specific information required to utilize the service endpoint.

C.1.1.1 Priority and Weight

The DNS SRV RR specification [RFC 2782] defines two fields, `Priority` and `Weight`, which provide for load-balancing and fault-tolerance. The SRV RR specification states that SRV RRs should be used in the order of their `Priority` and that requests should be delegated to services of equal priority, in accordance to the ratio of the SRV RR `Weight` value in relation to the SRV RRs of the same name and priority. For example, if the following services were configured in the DNS name server:

```
_http._tcp      IN  SRV  0  2  80    www
                IN  SRV  0  1  80    www2
                IN  SRV  1  1  8000  www3
```

The client should utilize the `www` and `www2` web servers before using the `www3` web server and the client should sent 2 requests to the web server named `www` for every 1 request sent to the web server named `www2`. The `www3` should only be used if the web servers name `www` and `www2` are not available. Please note, in this example all of the web servers are expected to contain identical web sites, i.e. identical copies of one another.

The use of the SRV RR in this manner is only useful if you have one web site per domain or one identical set of services per domain, however in reality this is rarely the case. To resolve this issue DNS-SD redefines the manner by which SRV RRs are named and adds TXT RRs for each SRV RR, please refer to Section C.2 - DNS SD Design, Implementation and Examples for more information regarding the changes made to the SRV RR definition to allow any service endpoint to be exposed via DNS-SD. The purpose and functionality of the SRV `Priority` and `Weight` values remains the same whether using the standard DNS definition or the DNS-SD definition of the SRV RR.

C.2 DNS SD Design, Implementation and Examples

DNS-SD addresses the issues presented in Annex C.1.1 above, by altering the use of the SRV Resource Record so that each SRV Resource Record has an instance specific name, such as `Internet\ Web\ Page._http._tcp`, which represents a single service endpoint for the Internet Web Page. Like traditional SRV Resource Records, there may be more than one SRV Resource Record for each instance name / service endpoint; when this occurs the SRV Resource Record's `Priority` and `Weight` values are used in the same manner as described for traditional SRV Resource Records [RFC 2782]. In order for a DNS-SD client to locate service instances of a particular service type, DNS PTR Resource Records (PTR RRs) are used. For example a PTR Resource Records named `_http._tcp` would point to a service instance name of a particular service type, which will then be characterized by the SRV and TXT Resource Records. The

following example demonstrates how DNS Resource Records are configured to implement DNS-Based Service Discovery.

```
_http._tcp                                  IN  PTR  New\ User\ Welcome\ Page._http._tcp
                                            IN  PTR  Intranet\ Web\ Site._http._tcp
                                            IN  PTR  Internet\ Web\ Site._http._tcp
New\ User\ Welcome\ Page._http._tcp         IN  SRV 0 1 80 intranet
                                            IN  TXT "path=/welcome.html"
Intranet\ Web\ Site._http._tcp              IN  SRV 0 1 80 intranet
                                            IN  TXT "path=/"
Internet\ Web\ Site._http._tcp              IN  SRV 0 1 80 www
                                            IN  TXT "path=/"
```

In this example, a DNS resolver queries the DNS name server for PTR Resource Records (PTR RRs) with the service type `_http._tcp.<domain>`. The DNS name server returns three PTR RRs, containing the service instances named `New User Welcome Page._http._tcp`, `Intranet Web Site._http._tcp`, and `Internet Web Site._http._tcp`. The client then queries the DNS name server for the SRV and TXT RRs for each service instance name retrieved from the the PTR RRs. The SRV and TXT RRs contain the service instance information, including the hostname, IP port and the additional service specific attributes, required to access the service endpoint.

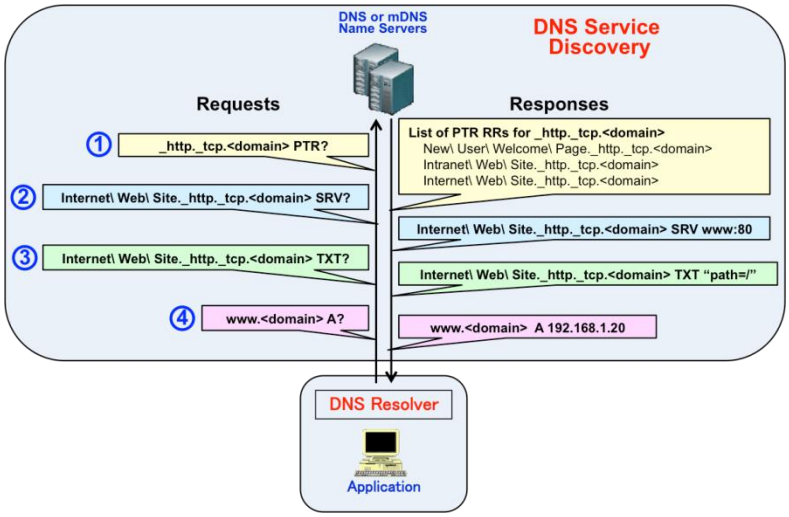

Figure C.2 – DNS-Based Service Discovery

As demonstrated in Figure C.2 and the example above, each SRV RR is accompanied by a TXT RR. The TXT RR contains additional service specific information that is required by the DNS-SD client to access the service instance endpoint. In this example only the URL `path` attribute is specified.

C.3 DNS-Based Service Discovery (DNS-SD) DNS Resource Record Types

As mentioned earlier in this section, DNS-SD uses existing DNS Resource Record types, but defines changes in the way SRV RRs are used and also defines the interaction between the SRV, PTR and TXT Resource Records. The following sections reiterate these changes and interactions.

C.3.1 DNS Resource Record CLASS Field

The CLASS field in the DNS Resource Records is a two-octet field used to indicate the protocol family of the DNS Resource Record. The CLASS allows DNS RRs to be grouped and queried by protocol family, such as "IN" for Internet or "CH" for Chaos. When using Multicast DNS (mDNS) [RFC 6762], the high order bit

(0x8000) of the CLASS field is used to indicate that a name resolution request may be responded to with a unicast UDP response, directed to the requestors IP address. This bit reservation reduces the available number of protocol families available to mDNS, but provides a means to prevent UDP broadcast storms.

C.3.2 DNS-SD Pointer (PTR) Resource Records

Pointer (PTR) Resource Records ([RFC 1035] and [RFC 2181]) are DNS Resource Records used by DNS-SD to associate one service type to one or more service instance names. Multiple PTR records may share the same service type, but each shall contain a unique combination of service type and service instance name.

C.3.2.1 Pointer Resource Record Format

The format of the DNS-SD PTR Resource Record [RFC 6763] shall be in the form:

```
Service TTL Class "PTR" Instance.Service
```

`Service` The symbolic name of the service type.

`TTL` Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the number of seconds that this RR can be cached before the information should be requeried from the source. Note that a value of '0' indicates that the RR should not be cached and shall be discarded immediately. (Optional)

`Class` Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to "IN", without the quotes, if unspecified. (Optional)

`"PTR"` Indicates that the RR is a Pointer RR. The value shall be equal to "PTR", without the quotes.

`Instance` Shall contain the human readable name of the service instance to which this PTR RR pertains.

Configuration Example:

```
_http._tcp           IN  PTR  New\ User\ Welcome\ Page._http._tcp
                     IN  PTR  Intranet\ Web\ Site._http._tcp
                     IN  PTR  Internet\ Web\ Site._http._tcp
```

A query issued by the resolver (`nslookup`) would have the following response from the name server.

```
% nslookup -type=ptr _http._tcp.example.org
Server:  my.nameserver
Address:  192.168.0.1

_http._tcp.example.org    Name = New User Welcome Page._http._tcp.example.org
_http._tcp.example.org    Name = Intranet Web Site._http._tcp.example.org
_http._tcp.example.org    Name = Internet Web Site._http._tcp.example.org
```

Where `nslookup` were executed with the argument "_http._tcp.example.org" and a type option specifying a query for the PTR RR type. The response from the name server shows that there are three PTR Resource Records associated to the service type "_http._tcp":

- `New User Welcome Page._http._tcp.example.org`
- `Intranet Web Site._http._tcp.example.org`
- `Internet Web Site._http._tcp.example.org`

C.3.3 DNS-SD Service (SRV) Resource Record

Service (SRV) Resource Records are used to define the location, i.e. the hostname and the port number, of a service instance. The SRV RR can be used to represent any service that runs on a specific port, but by itself does not support multiple services running on the same port. For DNS-SD, a TXT RR must accompany each SRV RR, even if that TXT RR is empty, containing no text.

C.3.3.1 Service Resource Record Format

The format of the DNS-SD SRV Resource Record [RFC 6763] shall be in the form:

`Instance.Service TTL Class SRV Priority Weight Port Target`

`Instance` Shall contain human readable name of the service instance to which this Resource Record pertains.

`Service` The symbolic name of the service type.

`TTL` Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the number of seconds that this RR can be cached before the information should be requeried from the source. Note that a value of '0' indicates that the RR should not be cached and shall be discarded immediately. (Optional)

`Class` Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to "IN", without the quotes, if unspecified. (Optional)

`"SRV"` Indicates that this RR is a Service RR. The value shall be equal to "SRV", without the quotes.

`Priority` Shall contain the Priority of this Resource Record relative to other SRV Resource Records sharing the same service instance name. Zero (0) is the highest priority.

`Weight` Shall specify the weight of the SRV RR, relative to other SRV RRs for the same service instance, of the same priority.

`Port` Shall contain the value of the Internet Protocol Port used by the service instance endpoint on the Target host.

`Target` Shall contain the domain name of target host.

Configuration Example:

```
New\ User\ Welcome\ Page._http._tcp      IN  SRV 0 1 80 intranet
                                         IN  TXT "path=/welcome.html"
Intranet\ Web\ Site._http._tcp           IN  SRV 0 1 80 intranet
                                         IN  TXT "path=/"
Internet\ Web\ Site._http._tcp           IN  SRV 0 1 80 www
                                         IN  TXT "path=/"
```

A query issued by the resolver (`nslookup`) would have the following response from the name server.

```
% nslookup -type=srv Intranet\ Web\ Site._http._tcp.example.org
Server:  my.nameserver
Address:  192.168.0.1

Intranet Web Site._http._tcp.example.org    SRV service location:
priority       = 0
weight         = 1
port           = 80
srv hostname   = intranet.example.org  internet address = 192.168.0.21
```

Where `nslookup` was executed with the argument "Intranet Web Site._http._tcp.example.org"and a type option specifying a query for the SRV RR type. The response from the name server shows that a HTTP service is exposed on the host `intranet.example.com` (`192.168.0.21`) on port `80`.

C.3.4 DNS-SD Text (TXT) Resource Record

Text (TXT) Resource Records ([RFC 1035] and [RFC 1464]) are DNS Resource Records used to associate textual attributes to each SRV Resource Record.

C.3.4.1 Text Resource Record Format

The format of the DNS-SD TXT Resource Record [RFC 6763] shall be in the form:

```
Instance.Service Class TTL "TXT" ( "AName=AValue" "AName=AValue" … )
```

`Instance`	Shall contain human readable name of the service instance to which this TXT RR pertains.
`Service`	The symbolic name of the service type.
`TTL`	Is an optional 32-bit unsigned integer ranging from 0 to 4294967295 (2^{32} - 1), representing the number of seconds that this RR can be cached before the information should be requeried from the source. Note that a value of '0' indicates that the RR should not be cached and shall be discarded immediately. (Optional)
`Class`	Two octets that shall contain one of the DNS Resource Record CLASS values, defaulting to "IN", without the quotes, if unspecified. (Optional)
`"TXT"`	Indicates that this RR is a Text RR. The value shall be equal to "TXT", without the quotes.
`AName`	Shall contain the unique attribute name.
`AValue`	Shall contain the value of the named attribute.

Configuration Example:

```
Intranet\ Web\ Site._http._tcp           IN  SRV 0 1 80 intranet
                                         IN  TXT "path=/"
```

A query issued by the resolver (`nslookup`) would have the following response from the name server.

```
% nslookup -type=txt Intranet\ Web\ Site._http._tcp.example.org
Server:  my.nameserver
Address:  192.168.0.1

Intranet Web Site._http._tcp.example.org  text = "path=/"
```

Where `nslookup` were executed with the argument "Intranet Web Site._http._tcp.example.org" and a type option specifying a query for the TXT RR type. The response from the name server shows that the text string `"path=/"` is associated with the service instance name `Intranet Web Site._http._tcp.example.org`.

C.3.5 Example DNS-Based Service Discovery Using “nslookup”

Being a simple command line tool, nslookup as a DNS Resolver allows one to manually test the behavior of DNS-Based Service Discovery (DNS-SD) from the command line. The example reproduced below is such an example. Using the configuration examples earlier in this section, the following example demonstrates how nslookup can be used to manually verify or test a DNS-SD configuration or to manually search for services.

```
% nslookup -type=ptr _http._tcp.example.org
Server:  my.nameserver
Address:  192.168.0.1

_http._tcp.example.org    Name = New User Welcome Page._http._tcp.example.org
_http._tcp.example.org    Name = Intranet Web Site._http._tcp.example.org
_http._tcp.example.org    Name = Internet Web Site._http._tcp.example.org

% nslookup -type=srv Intranet Web Site._http._tcp.example.org
Server:  my.nameserver
Address:  192.168.0.1

Intranet Web Site._http._tcp.example.org    SRV service location:
priority       = 0
weight         = 1
port           = 80
srv hostname   = intranet.example.org  internet address = 192.168.0.21

% nslookup -type=txt Intranet Web Site._http._tcp.example.org
Server:  my.nameserver
Address:  192.168.0.1

Intranet Web Site._http._tcp.example.org  text = "path=/"
```