10

DNS APPLICATIONS AND RESOURCE RECORDS

10.1 INTRODUCTION

DNS inherently lends itself well to "translating" a given piece of information into another related piece of information. This resolution process is the very reason for DNS's invention, and it has been extended beyond resolving hostnames into IP addresses and vice versa to support a broad variety of applications. Virtually any service or application that requires translation of one form of information into another can leverage DNS.

Each resource record configured in DNS enables this lookup function, returning a resolution answer for a given query. The DNS server parses the query from the Question section of the DNS message,* seeking a match within the corresponding domain's zone file for the query's QNAME, QCLASS, and QTYPE. Each resource record has a Name (aka Owner) field, Class (Internet class is assumed if not specified), and Type field. The RData field contains the corresponding answer to the query. The resource record type defines the type and format of the question (owner/name field) and corresponding answer (RData field). In some instances, multiple resource records may match the queried name, type, and class. In such cases, all matching records, called a *Resource Record Set* (*RRSet*), are returned in the Answer section of the response message.

* Refer to Figure 9.12.

IP Address Management: Principles and Practice, by Timothy Rooney

Most, but not all, new applications require new resource record types to enable definition of application-specific information, and these new resource record types are standardized via the IETF RFC process. This chapter describes the various forms of information that are stored in DNS along with the applications they support. A resource record summary is provided at the end of the chapter for reference.

10.1.1 Resource Record Format

First let's review the format of a resource record. When responding to a query for information, a DNS server will place the resource record information in the Answer section of a DNS message. The "on-the-wire format" dictated by the DNS protocol was introduced in Figure 9.13 in the context of the format of the DNS message Answer section, and is reproduced here as Figure 10.1 for convenience.

When representing resource records in zone files, all of these fields may be entered except the RDLength field, which is inserted when the resource record information is placed in a DNS message by the DNS server. The textual representation of a resource record generally follows a common convention shown below. Most resource records are defined with the following general fields, though many have subfields within the RData field as we shall see later in this chapter!

Owner	Time to Live	Class	Type	RData

Owner (Name). This field matches the information being queried.

Time to Live(TTL). The number of seconds for which the information contained in this resource record is valid for servers and resolvers caching this information. After the TTL expires, the resource record information must be removed from the name server and resolver cache. The TTL can be specified on a per resource record basis or if omitted, a zone level default TTL value is used ($TTL).

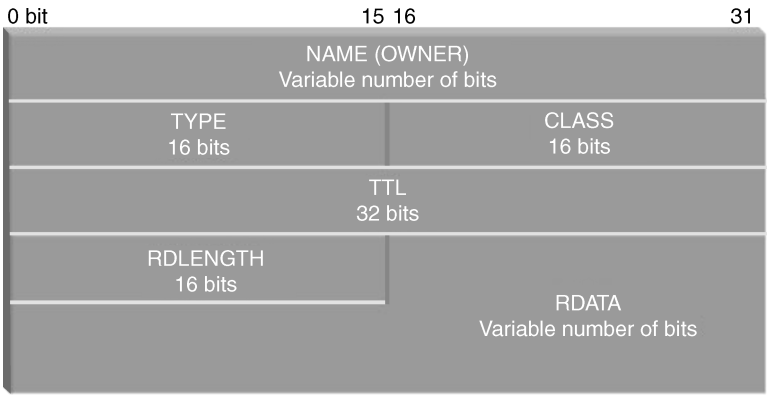


Figure 10.1. DNS resource record wire format (99).

Class. The Class of the resource record, usually IN for Internet.

Type. The type of resource record corresponding to the type of information being sought.

RData. The "record data" or answer portion corresponding to the information being sought by matching the Owner (Name), class, and type field contents.

Now that we've covered the basic format, we're ready to jump into specific applications and the resource records that support them. As we review these resource record types, we'll review the interpretation of each type and provide an example. We'll cover those that have been "officially" accepted by the IETF, that is, they've been published in an RFC; however, publication as an RFC does not guarantee universal implementation of the resource record type across all resolvers and servers. We'll point out some of those that may be new or experimental versus those ol' reliables that have been around for years.

For each record type we'll discuss in this chapter, the resource record fields and examples are displayed using a common format. The first row or table header, specifies the base fields defined above for each record type. The second row displays the interpretation of these base fields for the particular type in question. An example resource record of the given type is displayed in the third row and optionally successive rows as summarized below.

Resource record fields
Resource record field data types
Sample resource record(s)

Note that we will use the term "domain name" to refer to the name of a DNS domain, while the term "host domain name" will refer to the DNS name of a host. The host domain name may be defined with the zone file as fully qualified (FQDN) or simply a hostname interpreted in the context of the "current domain." The current domain is that defined in the zone declaration of the named.conf file, unless otherwise changed within the zone file using a $ORIGIN statement.

10.2 NAME–ADDRESS LOOKUP APPLICATIONS

10.2.1 Hostname and IP Address Resolution

First and foremost, the most common DNS application is hostname resolution, looking up a host domain name and obtaining its corresponding IP address. Two resource record types are supported for IP address lookups, one for IPv4 and the other for IPv6 addresses. The corresponding reverse record utilizes a common record type for both IPv4 and IPv6, the Pointer (PTR) record type.

When managing a mixed IPv4–IPv6 network, note that DNS will strongly influence which protocol will be used to reach a given destination host. For example, if I wish to access a web site, my resolver may first attempt to retrieve an A record for the given web

site address. Upon failing to obtain an IPv4 address, it may then attempt an AAAA record lookup with success. Assuming my browser (TCP/IP stack) supports IPv6, the connection will be made over IPv6. Unbeknownst to me, I'm using IPv6! Certain IPv4–IPv6 transition technologies explicitly force dual protocol lookups (A and AAAA) in DNS. We'll cover these technologies and the overall impact of DNS on IPv4–IPv6 networks in Chapter 15.

A—IPv4 Address Record. The A record is a common resource record type used to map a queried host domain name to an IPv4 address. The format follows the standard convention per the example below. Hosts may have multiple A records to provide load balancing or mapping of a hostname to multiple devices and/or interfaces.

Owner	TTL	Class	Type	RData
Host domain name	TTL	IN	A	IPv4 address
www.ipamworldwide.com.	86400	IN	A	10.100.0.99

AAAA—IPv6 Address Record. The AAAA ("quad-A") record provides an IPv6 address based on lookup of a host domain name. Formatted and processed similarly as the A record for hostname to IPv4 address lookup, the RData field includes an IPv6 address which can be abbreviated using standard IPv6 abbreviation conventions.

Owner	TTL	Class	Type	RData
Host domain name	TTL	IN	AAAA	IPv6 address
www.ipamworldwide.com.	86400	IN	AAAA	2001:DB8:3A::21:A450:1

PTR—Pointer Record. The PTR resource record provides mapping from an IP address to a FQDN. The PTR record is used to map both IPv4 and IPv6 addresses. The IPv4 version of the PTR includes the IP address reversed and concatenated with "in-addr.arpa." as the owner field and the corresponding FQDN as the RData field. The IPv6 version is formed by writing out the IPv6 address in its hexadecimal colon format, with all zeroes included; that is, fill in leading zeroes and double-colon shortcuts. Then drop the colons, reverse the digits, then concatenate with "ip6.arpa."

Owner	TTL	Class	Type	RData
IP address in reverse domain format	TTL	IN	PTR	Host domain name
1.32.65.10.in-addr.arpa.	86400	IN	PTR	sf1.ipamworldwide.com.
1.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0-.0.0.0.0. 0.0.0.0.0.0.8.B.D.0.1.0.0.2.ip6.arpa.	86400	IN	PTR	sf1.ipamworldwide.com.

The IPv4 address in this example corresponds to 10.65.32.1, while the IPv6 address is 2001:0DB8:0000:0000:0000:0000:0000:1001 or 2001:DB8::1001 in abbreviated form.

10.2.2 Alias Host and Domain Name Resolutions

The CNAME resource record type enables lookup of a host domain name by alias name. CNAME lookups return not an IP address, but a host domain name that must then be queried for its IP address, though most DNS servers responding to a CNAME query will include the corresponding A and/or AAAA record within the Additional section of the DNS response message. Meanwhile, the DNAME record provides a similar aliasing function for domains. As we discussed in the previous chapter, CNAME and DNAME records are useful in handling non-octet bounded reverse domains.

CNAME—Canonical Name Record. The CNAME record enables creation of alias names for hosts. The owner field contains the alias name being looked up, and the RData field yields the canonical host domain name. This host domain name would then need to be resolved to obtain the host's corresponding A and/or AAAA record.

Owner	TTL	Class	Type	RData
Alias host domain name	TTL	IN	CNAME	Canonical host domain name
w3.ipamww.com.	86400	IN	CNAME	www.ipamww.com.

Note that it is not legal to configure a CNAME RData field as pointing to another CNAME owner field in order to chain records. This RData field must point directly to an A/AAAA resource record owner name. The owner name of each CNAME record must also be unique; a single alias cannot resolve to multiple answers. CNAME records prove instrumental for mapping reverse domains as we discussed in Chapter 9.

DNAME—Domain Alias Record. The DNAME resource record, defined in RFC 2672 (107), enables mapping of an entire subtree of the domain name space to another domain. The major motivation for developing the DNAME record was to simplify DNS impacts of IP network renumbering. For example, if the company running the ipamww.com domain was acquired by acquired.com, the ipamww.com namespace could conceivably be "moved" beneath acquired.com with the addition of a DNAME record illustrated below.

Owner	TTL	Class	Type	RData
Alias domain name	TTL	IN	DNAME	Target domain name
ipamww.com.	86400	IN	DNAME	ipamww.acquired.com.

Resolvers seeking hosts within the ipamww.com domain subtree would be directed to seek the same hostnames under ipamww.acquired.com domain. Note that this scenario requires the resource records and subdomains of ipamww.com to be ported to their corresponding zone files within the acquired.com domain subtree. RFC 2672 stipulates that the DNAME owner (ipamww.com. in this case) zone must not have any subdomains nor contain any resource records besides the DNAME and possibly CNAME resource records.

10.2.3 Network Services Location

IP devices booting on a network often need to find specific services for device initialization. While DHCP provides some level of service location via specification of certain option values such as TFTP server IP addresses, DNS provides a services location mechanism using the services location resource record type (SRV). The SRV record provides a means for non-DHCP clients or for clients seeking services after initialization to locate servers providing requested services.

If you've worked with Microsoft clients and domain controllers since the introduction of Windows 2000, you're probably very familiar with SRV records. When Windows domain controllers boot up, they perform a dynamic DNS (DDNS) update for their A and SRV records, enabling them to effectively advertise services availability. These records are also easily recognized by underscores within the owner field. The SRV record owner field is comprised of a concatenation of a particular service, which is available via a particular protocol (TCP or UDP), for a given domain. The service name is prefixed with an underscore, as is the protocol value. The underscores were added to eliminate collisions with valid domain names. While technically not a valid host domain name character per the DNS RFC 1035, Microsoft, and BIND servers can be configured to tolerate the underscore character via the check-names option parameter. While a common example of the use of SRV records, SRV records are certainly not limited to Windows applications, though adoption beyond Windows applications has been limited thus far.

SRV—Services Location Record. The SRV record is used to enable resolver clients to identify servers offering particular services such as LDAP, Kerberos, and others. This record is critical for Microsoft Windows clients in locating Windows Domain Controllers.

Owner	TTL	Class	Type	RData			
Service encoding	TTL	IN	SRV	Priority	Weight	Port	Target host domain name
_ldap._tcp.ipamww.com.	86400	IN	SRV	10	0	389	ldap.ipamww.com.

The owner field is comprised of a concatenation of a particular service, which is available via a particular protocol (TCP or UDP), for a given domain. The RData field includes a priority field, which instructs clients to use numerically lower priority targets when multiple SRV records are returned.

The weight field is used to further prioritize records with the same priority. The port is the TCP or UDP port number to use to access the given service and the target is the host domain name of the server running the specified service.

If not also returned as additional information by the DNS server, the client may request corresponding A or AAAA records for hosts specified as targets to complete the resolution process. A couple of examples follow.

```
_ldap._tcp.ipamww.com. 86400 IN SRV 10 5 389 ldapeast1.ipamww.com.
_ldap._tcp.ipamww.com. 86400 IN SRV 10 10 389 ldapeast2.ipamww.com.
_ldap._tcp.ipamww.com. 86400 IN SRV 20 1 389 ldapeast3.ipamww.com.
```

In the three sample SRV records above, the second record would be used first. It shares the lowest priority number (10) as the first record, but it has a higher priority field (10) than the first record (5). The third record would be used last, as it has a larger priority value despite its lower weight.

The port, 389 in the examples above, is the TCP or UDP port number of the given service and the target is the hostname of the server running the specified service. If not also returned in the Additional section of the message from the DNS server, the client may request corresponding A or AAAA records for hosts specified as respective targets to complete the resolution process. Semantically, we didn't need to spell out the owner fields on the second two records assuming they were listed sequentially within the zone file, but we did so to emphasize that these three records would be returned for an SRV query for _ldap._tcp.ipamworldwide.com.

AFSDB—DCE or AFS Server Record (Experimental). The AFSDB record was defined in RFC 1183 (108) and was intended to enable location of a server, particularly for AFS (a registered trademark of Transarc Corp. and originally Andrew File System) and for the Open Software Foundation's Distributed Computing Environment (DCE).

Owner	TTL	Class	Type	RData	
Cell domain name	TTL	IN	AFSDB	Subtype	Host domain name
ipamworldwide.com.	86400	IN	AFSDB	1	afsdb1.ipamworldwide.com.

The RData field consists of

- Subtype field, which identifies the AFS 3.0 volume location server for the cell domain name (subtype = 1) or the DCE directory services server for the given cell domain name (subtype = 2).
- Host domain name field identifies the server hostname.

The AFSDB resource record is not widely used, as the SRV resource record type provides generic server location functionality within DNS and in fact RFC 5864 (189) specifies the use of SRV records for AFS.

WKS—Well Known Service Record (Historic). This resource record type identifies the well-known services such as FTP, telnet, and others that are available on a particular IP address using a particular protocol (TCP or UDP) for a host. This record is not generally used, as the SRV record provides similar functionality.

Owner	TTL	Class	Type	RData		
Host domain name	TTL	IN	WKS	IPv4 address	Protocol	Services
server.ipamww.com.	86400	IN	WKS	10.0.199.35	TCP	SMTP FTP

10.2.4 Host and Textual Information Lookup

The TXT record is one of the workhorse resource record types, often used as an interim resource record in support of specific applications pending standardization and implementation. The TXT record enables lookup of a generic reference name, for example, a domain name, host domain name, or other owner values, and returning arbitrary textual information. Most recently, the TXT record has been used for interim support of DDNS update uniqueness checking (now the DHCID record type) and for spam-reducing applications (the SPF record type), both covered later in this chapter.

TXT—Text Record. The text record enables the association of up to 255 bytes of arbitrary binary data with a resource record. It has proven very versatile in providing interim support of new services.

Owner	TTL	Class	Type	RData
Reference name	TTL	IN	TXT	Arbitrary text data
txt.cfo.ipamww.com.	86400	IN	TXT	“CFO Office (610) 555-1212”

HINFO—Host Information Record. The RData field of the HINFO resource record enables lookup of a host’s processor and operating system.

Owner	TTL	Class	Type	RData	
Host domain name	TTL	IN	HINFO	CPU	Operating system
sf1.ipamww.com.	86400	IN	HINFO	VAX 770/11	UNIX

HIP—Host Identity Protocol Record (Experimental). The HIP resource record type supports the experimental host identity protocol (HIP), which essentially abstracts the association of a hostname with an IP address by inserting a “host identity” layer in the resolution process. This enables association of a domain name with a host identity, which is then associated with one or more IP addresses. An application or upper

layer protocol can look up a host via the HIP resource record and obtain the host identifier (in the form of a public key) and other host identity information, including the IP address of the host or of a rendezvous server through which to connect to mobile devices.

Owner	TTL	Class	Type	RData					
Host domain name	TTL	IN	HIP	HIT Len.	PK Alg.	PK Len.	HIT	Public key	RVS
hiphost.ipamww.com.	86400	IN	HIP	16	2	24	Iil...	8L9d...	rs.ipamww.com.

The RData fields are defined as

- *HIT Len*. Length in bytes of the Host Identity Tag (HIT); this field is inserted by the server for wire transmission and is not displayed within a zone file.
- *PK Alg*. The algorithm used to generate the Public key
  - 0 = no key is present
  - 1 = DSA formatted key
  - 2 = RSA formatted key
- *PK Len*. Length in bytes of the public key; this field is inserted by the server for wire transmission and is not displayed within a zone file.
- *HIT*. The Host Identity Tag, a 128-bit hash of the host identifier.
- *Public Key*. The public key associated with the host that can be used to validate signed messages from the host.
- *RVS (optional)*. One or more rendezvous server host domain name(s) for connecting with mobile devices.

RP—Responsible Person Record. The RP resource record enables association of an email address and other text information with a node in the domain tree, whether an end host or domain. The RData field contains an email address, formatted without the @ sign; instead, a dot is substituted for the @ sign. The second field of the RData field indicates a record for which additional text information can be found as an additional lookup.

Owner	TTL	Class	Type	RData	
Host domain name	TTL	IN	RP	Email address	TXT pointer
payroll.ipamww.com.	86400	IN	RP	cfo.ipamww.com.	cfo-contactinfo.ipamww.com.

In this example above, we've used an RP record to associate the payroll server with our CFO, reachable at cfo@ipamww.com (substitute "." for "@" in email address field). The

TXT pointer field points to a resource record containing additional information, such as the following example:

```
cfo-contactinfo.ipamww.com. 86400 IN TXT "CFO Office (610)-555-1212"
```

10.2.5 DNS Protocol Operational Record Types

Two "administrative" resource record types enable specification of zone authority information (the SOA record) and delegation name servers for this and child domains (NS). These record types are instrumental to the efficient operation of keeping DNS data in synch within a zone and of keeping delegation chains in effect down the domain tree.

SOA—Start of Authority Record. One and only one SOA record is required for each zone and follows the initial default TTL ($TTL) statement if present within the zone file. The SOA record defines the domain name for which this zone is authoritative, along with additional zone maintenance information. The SOA record is composed of the following fields.

Owner	TTL	Class	Type	RData						
Domain name	TTL	IN	SOA	mname	Contact	Serial number	Refresh interval	Retry interval	Expire interval	Negative cache
ipamww.com.	86400	IN	SOA	ns1.ipamww.com.	admin.ipamww.com.	3945	2h	30m	1w	1d

- Domain name for which this zone file contains authoritative information.
- TTL, time to live.
- Record class (IN for Internet).
- Record type (SOA).
- *Master DNS Server Name (MNAME).* The name of the DNS server that is master for this domain (zone).
- Domain contact email address (replace "@" with "." so that admin@ipamworldwide.com is written admin.ipamworldwide.com. Note that email addresses with dots prior to the @ sign should be prefixed with a backslash. Thus, super.admin@ipamww.com would be encoded as super\.admin.ipamww.com.
- *Serial Number of the Zone*. Incremented with every change to zone data—enables slave servers to identify changes to zone data.
- *Refresh Interval.* Time period for slaves to query the master for zone updates.
- *Retry Time*. If unable to reach the master, the slave will wait this amount of time to retry to reach the master.

- *Expire Time*. If unable to reach the master after this amount of time expires, the slaves will delete the zone information and no longer consider itself authoritative, thereby expiring its authority for the zone.
- *Negative Caching TTL*. Time duration to maintain cache of negative responses from other servers; for example, a specified domain or record doesn't exist.

An example SOA record for our ipamww.com zone file might look like

```
ipamww.com. IN SOA dns1.ipamww.com dnsadmin.ipamww.com (

    1    ; serial number
    2h   ; refresh interval of 2 hours
    30m  ; retry after 30 minutes
    1w   ; expire after 1 week
    1d)  ; negative caching TTL of 1 day.
```

NS—Name Server Record. The NS record enables lookup of an authoritative name server for a given zone. NS records are the key to distributing the DNS database. In delegating a child domain to another administrative authority, the child domain administrator must be running at least two name servers for redundancy. While traversing the domain tree, these NS records enable the queried name server along the resolution path in the domain tree to respond with a referral to another name server further down the tree, which has more information about the intended destination. Each zone must also declare at least two NS records for its authoritative name servers as well.

Owner	TTL	Class	Type	RData
Domain name	TTL	IN	NS	Name server domain name
ipamworldwide.com.	86400	IN	NS	ns1.ipamworldwide.com.

Note that the name server hostname in the RData field should have a corresponding A or AAAA record to complete the required resolution to a reachable IP address. This is referred to as a "glue" record in that it "glues" the resolution of the authoritative name server hostname for the desired domain to the IP address of that name server.

10.2.6 Dynamic DNS Update Uniqueness Validation

DHCID—Dynamic Host Configuration Identifier Record. Dynamic DNS enables the updating of DNS information with DHCP clients' assigned IP address information. Thus, a DHCP server on behalf of the client or the client itself can update DNS with the client's IP address and hostname association via A/AAAA and PTR* records. It is quite possible that the same hostname/FQDN may be claimed by multiple

* Associating client identification information with PTR records is not currently specified in the DHCID RFC.

DHCP clients, or that a client may claim a hostname already assigned to a predefined (e.g., statically addressed) device.

The DHCID record provides client identification information in DNS to uniquely associate the particular DHCP client with the hostname/FQDN being updated by the DHCP server. The DHCID record would be defined in the Prerequisite section of the DNS Update message to verify the record "owner" for updating. Please refer to the DNS Update section of the previous chapter for more details and an example of this prerequisite processing.

The DHCID record uses the same owner field as the corresponding A or AAAA record. The RData portion of the record is formed by performing a one-way secure hash using the SHA-256 algorithm over the following concatenated fields:

- *Identifier Type Code* (2 bytes). Identifies the information within the DHCP packet that was used in creating this hash. Possibilities include client hardware address, client identifier option, or device unique identifier (DUID).
- *Digest Type Code* (1 byte). Identifies the hash algorithm. The RFC defines values of 0 (reserved) or 1 (SHA-256) though IANA maintains a registry for future value assignments.
- Digest of the data from the DHCP packet as identified by the identifier value concatenated by the client's FQDN.

Owner	TTL	Class	Type	RData		
Host domain name	TTL	IN	DHCID	Identifier Type	Digest Type	SHA-256 hash of {identifier type, fqdn}
w3.ipamww.com.	86400	IN	DHCID	A1B87Y2/AuCcg8e93aQcjl. . .		

10.2.7 Telephone Number Resolution

DNS has proven very versatile and can even be used to map telephone numbers into IP addresses, which is useful for VoIP applications or related telephony over IP applications. The ENUM (E.164 telephone number mapping) service has been defined to support such resolution. ENUM supports the mapping of telephone numbers, in ITU E.164 format, into uniform resource identifiers (URIs).* This mapping is performed primarily using the Naming Authority Pointer (NAPTR) resource record type.

Note that most enterprise IP PBX systems provide their own directories to map intra-PBX phone numbers to destination phones' IP addresses, so ENUM is not commonly implemented in such environments. However, VoIP service providers have a vested

* A URI is an Internet identifier consisting of a uniform resource name (URN) and a uniform resource locator (URL). A simple example: for URL `http://ipamworldwide.com` and URN `file.txt`, the corresponding URI is `http://ipamworldwide.com/file.txt`.

interest in assuring calls remain on their or their partners' IP and access networks to the maximum extent, to reduce call handling costs paid to nonpartner network providers, or worse, competitors. And ENUM is key to enabling such call routing by virtue of telephone number mapping or resolution. That is not to say that you won't see ENUM within enterprise networks. ENUM provides resolution to multiple destinations with preference settings, which may find use within reachability or contact management type applications.

As just mentioned, the NAPTR resource record provides translation of telephone number information into destination uniform resource identifiers. Currently defined in RFC 3403 (109), NAPTR records were initially defined to provide a means to iteratively resolve an arbitrary string into a URI for the Dynamic Delegation Discovery System (DDDS). Some background on DDDS is provided in RFC 3402 (110), but it initially stemmed from the desire to define a resolution process that could enter with a resource name (e.g., a particular application or piece of data) which in itself, contains no network location information, and resolve it to a destination resource identifier by applying a series of iterative rules from a database. This separation of the specification of the resource name from the process to locate or resolve it facilitates the making of changes and redelegations of resources without impacting the end user application's naming convention.

This effort expanded beyond resolving resource names to supporting resolution of generic lookup strings, and evolved into the DDDS, using DNS as one form of the rules database. The NAPTR record enables the specification of such rules within DNS, sometimes using multiple NAPTR records to fully complete the resolution process. Each NAPTR record translates a given entry string, that is, a valid DNS domain name, into a rule that can be applied to the string to derive the next string to lookup. This process iterates until a terminal rule is reached and the final result is returned to the requesting application.

NAPTR records are the building blocks of E.164 telephone number mapping service for service provider voice over IP services. RFC 3761 (111) provides the "application specific" interpretation of NAPTR fields for the ENUM application. A NAPTR record can be used to lookup a destination telephone number, and resolve the number to a destination, for example, a Session Initiation Protocol (SIP) server, email address, or other URI-formatted destination. NAPTR records also support the ability to define regular expressions, which supply logical rules as "next steps" for the resolver to locate the intended destination.

E.164 is an International Telecommunications Union (ITU) standard for formatting telephone numbers. "Fully qualified" telephone numbers, meaning they are globally unique given the country code prefix followed by a country-specific telephone number format, are represented with a plus sign prefix, such as +1-610-555-1234. Much like reverse domains for IP addresses, formatting a telephone number requires a similar convention of reading the resource record from left to right as more specific to less specific. This convention requires reversal of the fully qualified telephone number (dropping the plus sign) and separating each digit with "dots" as illustrated below.

Note the use of the .arpa top-level domain. Similar to ip6.arpa and in-addr.arpa domain structures, the e164.arpa domain is a "reverse" domain in that it enables lookup

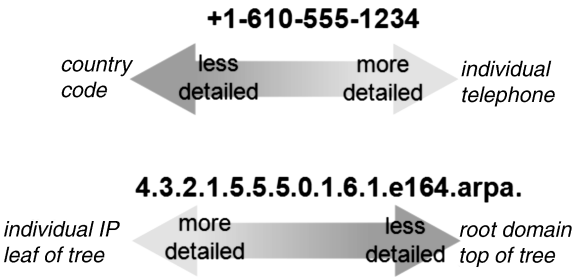


Figure 10.2. Telephone number mapping to domain structure.

of a structured numerical value, a phone number. Like other .arpa lookups, the domain structure is organized top to bottom as generalized-to-specific, or country code-to-telephone line number. Thus, the fully formatted E.164 telephone number is reversed, each digit is separated with dots, and the e164.arpa. domain suffix is appended as illustrated in Figure 10.2.

This structure lends itself well to segmentation of telephone number space. For example, the domain 1.e164.arpa refers to all country code 1 telephone numbers and could be delegated to such a number authority. Likewise 44.e164.arpa could be delegated to the U.K. telephone numbering authority. Within each of these domains, further delegation may be accomplished in accordance with the numbering plan for the country. For example, within the United States, an area code represents the next logical administrative delegation point, followed by exchange. Thus, the administrators for 1.e164.arpa may delegate the 0.1.6.1.e164.arpa zone to the numbering administrator for the 610 area code, who may in turn delegate 5.5.5.0.1.6.1.e164.arpa to those responsible for the 555 exchange within the 610 area code.

NAPTR—Naming Authority Pointer Record. The NAPTR record provides translation of a string* or telephone number information into destination uniform resource identifiers. The NAPTR record utilizes the e164.arpa. domain naming convention described above within its owner field to serve as the lookup format for telephone numbers. Unfortunately, this so far is only the easy part! The NAPTR record contains a number of additional subfields with its RData field. The additional subfields are described below, with examples provided for the ENUM application of NAPTR records.

- *Order Field.* Specifies the order in which multiple records within the RRSet are to be processed; lower numbered order records are processed first.
- *Preference Field.* Specifies the order in which records with equal "order" values are to be processed; lower numbered preference records are processed first.
- *Flags.* Provides information about the "next lookup" in the resolution process. Thus far, four flag values have been defined, though the Flags field can be empty:

* "Strings" refer to text or data strings. Fortunately, this is not the "string theory" of DNS!

- "u" The output of the regular expression field of this record is a uniform resource identifier; that is, this is a terminal resolution.
- "s" Next lookup should be for SRV records.
- "a" Next lookup should be for A, A6, or AAAA records.
- "p" Next lookup is protocol specific according to the protocol specified in the Services field.

- *Services*. This field encodes the services that are available based on the application in question. This field includes the type of resolution provided, a "+" sign or colon, followed by the protocol value, for example, http, sip, mailto, ftp, tel, among others.* Examples of types or resolution include
  - I2L. URI to URL.
  - N2L. Uniform Resource Name (URN) to URL.
  - E2U. ENUM service to URI.
- *Regular Expression*. An encoded expression that is to be evaluated. The syntax of this field is a sed-style expression.
- *Replacement*. An alternative "next lookup" fully qualified domain name in the absence of a regular expression.

Owner	TTL	Class	Type	RData					
Domain name	TTL	IN	NAPTR	Order	Pref	Flags	Services	Regexp	Replacement
me.ipamww.com.	86400	IN	NAPTR	10	5	"s"	"N2L + http"	" "	www.ipamww.com.
4.3.2.1.5.5.5.0.1.6.1.e164.arpa.	86400	IN	NAPTR	10	20	"u"	"E2U + sip"	"!^.*$!sip:me@ipamww.com.!"	

Let's look more closely at the two example NAPTR records above. The first example provides a rule for resolution of me.ipamww.com. The Flags field value of "s" indicates to the resolver that the next lookup should be a query for SRV resource records. The Services field indicates a URN-to-URL service using HTTP protocol. Since the regular expression field is blank, the replacement field is used as the result of the resolution process.

The second example highlights an ENUM application example, where a lookup of a telephone number can be resolved. The "u" flag indicates that the result of the regular expression provided will be a URI, which can then be resolved to an IP address. The Services field indicates ENUM services using the SIP protocol. The regular expression field is comprised of two subfields, encapsulated with the "!" character. The first field

* Please consult http://www.iana.org/assignments/enum-services for the currently assigned services values for ENUM.

contains "^.*$" and is interpreted as "match from the start of the line (^) to the end of the line ($), zero or more (*) characters (.)"; that is, match the entire Owner field. The second portion of the regular expression contains "sip:me@ipamworldwide.com" which is returned as the result of our regular expression. The Replacement field is not used in this case.

The resulting URI, sip:me@ipamworldwide.com would then initiate a DNS query for an address (A or AAAA) record for ipamworldwide.com. Note that some DNS servers may return relevant A or AAAA records as additional information in the query response containing the NAPTR records. The resulting IP address would be used as the destination address to initiate the sip session to the "me" user.

10.3 EMAIL AND ANTISPAM MANAGEMENT

Spam email or unsolicited bulk email, has been a nuisance since the dawn of the Internet, though in the early days it was highly frowned upon. Nevertheless, with the explosive growth of the Internet, the volume of spam emails has seemingly grown even faster. A variety of techniques exist to combat spam, many of which involve the use of DNS. To understand how DNS can help reduce spam, we'll first look at the anatomy of an email transmission including the role of DNS in email delivery, then review the use of DNS in various antispamming solutions.

10.3.1 Email and DNS

An email typically originates from one person and is sent to one or more recipients. Each email address is formatted as a mailbox@maildomain. The mailbox commonly refers to the name of the person or owner of a mailbox or email account, while the maildomain, typically the company or Internet provider name, is the destination domain for delivery to the corresponding mailbox or mail exchanger. Emails are composed using an email client, such as Microsoft Outlook, Eudora, or web-based clients such as yahoo and google. Regardless, when sent by the originator, the client connects to a Simple Mail Transfer Protocol (SMTP) server (using the SMTP protocol) to send the email. Like a default router for email, the SMTP server is responsible for forwarding the email to its destination.

The SMTP server must resolve the maildomain to an IP address for transmission of the message. Naturally this is done using DNS with a lookup for the mail exchanger (MX) record type, as well as the corresponding A or AAAA record types.

MX—Mail Exchanger Record. The mail exchanger record is used to locate an email server or servers for a particular domain. If I send an email destined to tim@ipamworldwide.com, my SMTP server will use DNS to find the host(s) that can receive emails for users in the ipamworldwide.com domain. More than one MX record may be created per domain, and each can be defined with a different preference value. Use of the preference field enables the sending SMTP server to prioritize the destination host to which it will forward the email for the given domain, and if unavailable to a second (and

third, etc.) choice destination. The lower the preference value, the more preferred the listed destination. In the example below, we have two MX records for the ipamworldwide.com domain. The destination smtp1 is preferred (lower preference) over smtp2. However, if smtp1 is unavailable, this mechanism provides a backup server for email delivery.

Owner	TTL	Class	Type	RData	
Email destination domain	TTL	IN	MX	Preference	Mail server host domain name
ipamworldwide.com.	86400	IN	MX	10	smtp1.ipamworldwide.com.
ipamworldwide.com.	86400	IN	MX	20	smtp2.ipamworldwide.com.

Note that the mail server host domain name within the RData field must have a corresponding A or AAAA record to complete the required resolution to a reachable IP address. Many DNS servers supply these address records within the Additional section of the MX query response.

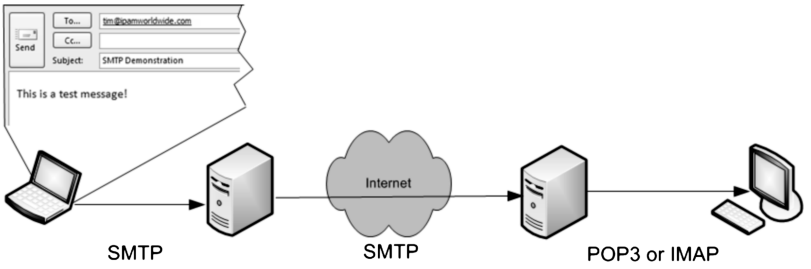
Upon resolving the destination mail server, the SMTP server sends the message to the destination using the SMTP protocol. The ultimate destination server, to which recipient email clients connect, must support Post Office Protocol (POP) or Internet Message Access Protocol (IMAP) to enable client retrieval of the email message. Thus, when your email client performs a "send/receive," it utilizes SMTP to send outgoing messages to its configured SMTP server and POP or IMAP to retrieve incoming email messages from the configured POP/IMAP server(s).

Figure 10.3 highlights a very simple SMTP transaction between two servers, when my friend Mike sends me an email. On the left of the figure, Mike composes an email to tim@ipamworldwide.com using his email client and sends it. His configured SMTP server forwards the message to the destination server, as resolved by the MX record(s) for ipamworldwide.com. His SMTP server initiates a TCP connection on port 25 with the resolved destination server.

Once the TCP session is established, the SMTP application utilizes the session to handshake and process the message. The envelope portion of the message begins with the HELO (or EHLO, enhanced HELO), which conveys the sending entity's identity. The MAIL FROM statement indicates the source of the message, followed by the RCPT TO statement indicating the destination mailbox. At this point in the exchange, the recipient server may refuse to accept the message and close the connection if the destination mailbox is unknown or blocked, or if the "from address" is prohibited. Otherwise, the transaction continues and the data or message portion* is transmitted. The receiving mail exchanger stores the email message or forwards it to the server on which the destination mailbox resides.

The store-and-forward approach used by the received email server may also be used by intermediate email gateways (aka message transfer agents) to provide multihopped email delivery. As mentioned above, the resolution of a destination mailbox domain to

* Note that the message portion of an email consists of a header and the body. As a point of reference, RFC 2821 (163) defines the SMTP specification, while RFC 2822 (164) defines the Internet message format for email, defining valid header and data syntax.



TCP Setup

Envelope
- ◄ 220 smtp.ipamworldwide.com
- ► EHLO smtp.ipamww.com
- ◄ 250 ipamww.com
- ► MAIL FROM:<mike@ipamww.com>
- ◄ 250 OK
- ► RCPT TO:<tim@ipamworldwide.com>
- ◄ 250 OK

Message
- ► DATA
- ◄ 354 Start mail input; end with <CR><LF>.<CR><LF>
- ► Subject: SMTP Demonstration
- ► From: mike@ipamww.com
- ► To: tim@ipamworldwide.com
- ► This is a test message!
- ► .
- ◄ 250 OK

QUIT
221 BYTES

Figure 10.3. Simple SMTP transaction example.

multiple MX records implies this ability to identify a "destination" mail server, which may or may not be the final destination from which the intended recipient retrieves the email. The MX record preference field provides control over the relative preference of incoming mail servers or gateways, while providing selection from among multiple choices based on availability and performance.

Figure 10.4 illustrates a two-step email delivery scenario using SMTP. In this scenario, I'm sending the same email as shown in Figure 10.3. However, in this case, perhaps the intended destination server, smtp.ipamworldwide.com is busy and refuses a direct connection. Having resolved both the ipamworldwide.com server and an mta-gateway.com server via a DNS MX query, my outgoing mail server will attempt to send the email to the second choice, mta-gateway.com.

In accepting the SMTP transmission from my mail server, the mta-gateway.com server effectively agrees to forward the email to the ultimate destination on my behalf. The transaction between my mail server and the mta-gateway.com server completes before the second leg of transmission is attempted. SMTP uses a store-and-forward approach, not synchronous relaying of each message.

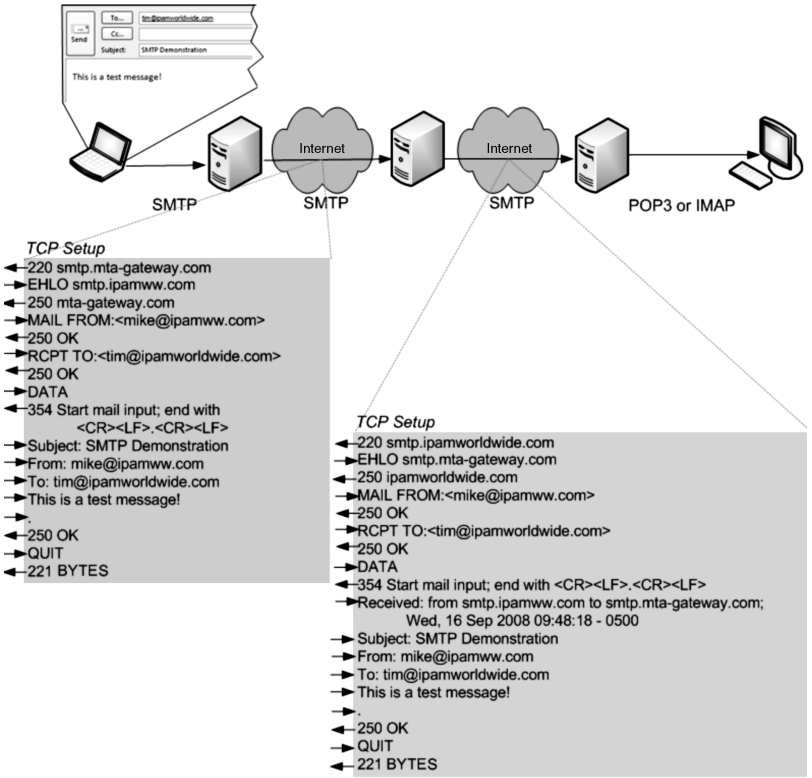


Figure 10.4. Email relay.

The first leg of the transmission looks very similar to that of Figure 10.3, except for the difference in the SMTP server. The second leg of the connection is also similar, except once again for the SMTP endpoints. The other difference is the insertion of the *Received:* line within the header portion of the data section of the mail. Each intermediate SMTP server which forwards the message prefixes a "Received" line indicating its domain name and corresponding time stamp. This enables tracing of the email from the destination back to its path. The RCPT TO line remains the same in both segments, indicating the mailbox to which errors in delivery should be sent.

As footnoted above, the message portion of an email consists of a header and the body. Each header field consists of a word followed by a colon and a value. The header contains a variety of data including the following:

- *Originator Fields*. From, sender, reply-to, orig-date;
- *Destination Fields*. to, cc, bcc;
- *Identification Fields*. Message-id, in-reply-to, references, msg-id, id-left, id-right, no-fold-quote, no-fold-literal;
- *Informational Fields*. Subject, comments, keywords;

- *Resent Fields* (informational fields relating to the reintroduction* of a message into the Internet, for example, by an emailing service). Resent-date, resent-from, resent-sender, resent-to, resent-cc, resent-bcc, resent-msg-id; and
- *Source Trace Information*. Trace, return, path received, name-val-list, name-val-pair, item-name, item-value.

We have summarized the basic email process and types of information that may be included in a given email message because different antispam techniques utilize different information sources in validating the sender as a legitimate or acceptable sender of emails. We'll discuss those techniques that utilize DNS to perform this validation next.

10.3.2 White or Black Listing

The use of white or black listing (190) provides a simple means for the recipient email server to lookup a sender's IP address via DNS and to validate its legitimacy. This lookup is typically formed by reversing the IP address of the source IP address of the email message, just as is done in forming PTR records. Note that the source IP address being analyzed is that from which the email was received directly, perhaps an email gateway, which may or may not be the original transmitter. However, the intent of such listing is to identify such senders of email by IP address as legitimate or not.

In this scenario, the reversed IP address is appended with a given domain name, typically that of the black list provider. The "host domain name" thus formed by this concatenation is queried in DNS using the A resource record query type, not PTR. The query answer is interpreted based on whether the record was found, in which case often an IP address within the 127/8 block is returned, and on whether the list publishes known spammers (black or block list) or known nonspammers (white list).

For example, upon receiving an email message with a source IP address of 192.0.2.95, my email server formulates an A record query for hostname 95.2.0.192.spamblacklist.org, assuming my chosen black list provider publishes lookups within the spamblacklist.org domain. Upon receiving a reply with answer (IP address) 127.0.0.5, my email server classifies the email as spam and rejects it. On the other hand, if NXDOMAIN is returned for the query, the email may be permitted. A white list service, publishing known genuine email server addresses would render the opposite interpretation based on the DNS lookup.

10.3.3 Sender Policy Framework (SPF)

The Sender Policy Framework is currently defined with RFC 4408 (112) under experimental status. SPF enables an organization to publish its own list of authorized outgoing email server addresses, a self-published white list, though with substantially more sophistication. Under SPF, the received email message's envelope information is

* Reintroduction is not forwarding. The transmission of an email with the *original sender* information instead of that of the transmitter is considered reintroduction. Forwarding uses the mailbox doing the forwarding as the sender.

examined, and an SPF DNS query from the recipient is based upon the sender, the sender's domain, as well as the source IP address. Upon receipt of an email message, the recipient email server would issue a query for an SPF resource record for the source domain name. The SPF record is encoded as a string of "mechanisms" that are used to process the source IP address from which the email originated, the domain portion of the MAIL FROM or HELO identity, and the sender from the MAIL FROM or HELO identity.

SPF—Sender Policy Framework Record. The Sender Policy Framework attempts to provide validation of what hosts are configured to send email for a given domain. That is, SPF seeks to eliminate spam emails from spoofed domains. A recipient email host can look up the SPF records for the sender's domain to verify that the sending email host matches those authorized by the sender. SPF version 1 or SPF classic as it is also called, is documented in RFC 4408 and utilizes the SPF resource record. Domain administrators can configure DNS with email hosts mapping to each host's mailfrom and SMTP HELO identities. SenderID is a related spam detection technique that also uses the SPF resource record type, though it analyzes different information from an incoming email message. We'll cover SenderID a bit later.

Note that due to actual implementations of SPF using TXT records prior to IETF publication of RFC 4408, most implementations will use both SPF and TXT records for backward compatibility, though an SPF compliant resolver will discard the TXT records if both TXT and SPF records are returned. The format of the SPF record is identical to that of the TXT record; however, a particular syntax is employed for SPF applications instead of arbitrary text. The syntax includes a version string (`v=spf1` for SPF, `spf2.0` for SenderID covered next) followed by a space, then one or more terms that define qualifiers on resource record types or IP network addresses, modifiers, and even macros.

Owner	TTL	Class	Type	RData
Domain name	TTL	IN	SPF	Version, directives, and/or modifiers
smtp.ipamww.com.	86400	IN	SPF	v=spf1 + ip4:192.0.2.32/30 –all
smtp.ipamww.com.	86400	IN	SPF	spf2.0 pra + ip4:192.0.2.32/30 –all

Mechanisms. Mechanisms enable specification of the match criteria within the SPF (or TXT) record, which a receiving email server can query to validate the sender of a given email message. Mechanisms are defined within the SPF record's RData field after specification of the SPF version, currently version 1, "v=spf1." Mechanisms are evaluated left to right. If a mechanism passes based on evaluation of the mechanism, the verification passes; otherwise, the next mechanism is tested until a pass or fail is found or no further mechanisms are defined.

Each mechanism can be defined with a qualifier, a prefix that instructs the mail or spam filter server how to interpret a given "match":

- + = pass (default). Consider this mechanism a pass, if this mechanism matches.
- - = fail. Consider this mechanism a fail, if the mechanism matches.

- ~ = soft-fail. Consider this mechanism somewhere between neutral and fail, if this mechanism matches; this interpretation would not fail this check outright if it matched, but would hold it for closer scrutiny.
- ? = neutral. Consider this mechanism neutral, if this mechanism matches.

Qualifiers may be used with the following resource record check based mechanisms to define the interpretation of a given mechanism as shown in the examples following:

- `a` = lookup the A record for the source domain (from the MAIL FROM or HELO identity); if it matches the source IP address of the message, this mechanism matches. This can be scoped to a specific domain and/or number of CIDR bits to compare in the addresses as illustrated in the following examples:
  - `+a` = pass, if the A record query for the source domain matches the source IP address.
  - `-a:ipamworldwide.com` = fail, if an A record query for ipamworldwide.com matches the source IP address.
  - `~a/24` soft-fail, if the first 24 bits of the IP address retrieved via A record lookup of the source domain matches the first 24 bits of the source IP address.
- `mx` = lookup the MX record for the source domain (from the MAIL FROM or HELO identity); for each MX lookup resolved, look up the corresponding A record; if it matches the source IP address of the message, this mechanism passes. As with the `a` mechanism, the `mx` mechanism can be scoped to a specific domain and/or number of CIDR bits to compare in the addresses as illustrated in the following example:
  - `+mx:ipamworldwide.com/28` = pass, if an A record associated with a MX record lookup is returned where the first 28 bits match the first 28 bits of the source IP address of the message.
- `ptr` = lookup the PTR record (up to 10) corresponding to the source IP address of the email message; then compare two things with each domain name returned in the PTR lookup:
  - Check that the domain name returned matches the source domain of the email message.
  - Check that the corresponding A or AAAA record returns an IP address matching the source IP address.

  If both conditions hold, this mechanism passes. This mechanism can be further scoped by a domain name, which can be used to filter multiple returned PTR-lookup domain names as illustrated in the following examples:
  - `-ptr:` fail, if a domain name returned during the PTR lookup of the source IP address matches the source domain and if the A/AAAA domain name corresponding to the domain name returned during the PTR lookup matches the source IP address of the email.

  - `+ptr:ipamworldwide.com`: pass, if a domain name returned during the PTR lookup of the source IP address matches the source domain while falling within the ipamworldwide.com domain and if the A/AAAA domain name corresponding to the domain name returned during the PTR lookup matches the source IP address of the email.
- `ip4` = verify that the source IP address matches the IPv4 address specified; this mechanism may be qualified by CIDR length as illustrated in the following example:
  - `?ip4:192.0.2.32/30`. Neutral, if the source IP address of the message falls within 192.0.2.32-192.0.2.35.
- `ip6` = verify that the source IP address matches the IPv6 address specified; this mechanism may be qualified by prefix length as illustrated in the following example:
  - `+ip6:2001:db8:f02b:2a::/64`. Pass, if the source IP address of the message falls within the 2001:DB8:F02B:2A::/64 network.
- `exists:`*`domain_name`* = lookup the A record (not AAAA) corresponding to the *domain_name*; this mechanism matches if any answer (IP address) is provided (this mechanism must be scoped with a domain name to match as illustrated in the following example).
  - `exists:ipamworldwide.com`: matches, if an A record lookup for the ipamworldwise.com domain returns an IP address.
- `include:`*`domain_name`* = recursively evaluate the domain_name to leverage its SPF policies, for example, to utilize the policy of a domain from multiple ISPs or from other domains from which you send email.
- `all` = matches everything; often used as the final parameter as `–all` to fail if no prior mechanism matches.

Modifiers. Modifiers may be specified within SPF records to provide additional information. Modifiers are name-value pairs, two of which have yet been defined:

- `redirect` = *`domain_name`*: enables "aliasing" of SPF records, for example, to apply a common SPF processing record to multiple domains. This provides a convenience for ongoing change management: change the processing in one record, minimizing errors, and maximizing consistency. In the following example, the MX record check for the ipamworldwide.com domain would apply to the hq and euro subdomains as well.

```
hq.ipamww.com. IN SPF ''v=spf1 redirect=_spf.ipamworldwide.com''
euro.ipamww.com. IN SPF ''v=spf1 redirect=_spf.ipamworldwide.com''
_spf.ipamworldwide.com. IN SPF ''v=spf1 +mx:ipamworldwide.com –all''
```

The redirect can be used explicitly as in the above example, or as a "last resort", for example, listed as the rightmost mechanism.

- `exp` = *`domain_name`*: explanation, which defines the domain for which a TXT record lookup must be done to identify the string to be presented as results upon a mechanism match failure.

Macros. Technically, the *domain_name* for any of the above mechanisms and modifiers need not be an explicitly defined (hard coded) domain, but one that can be defined using macros to dynamically formulate a domain name based on the message envelope under evaluation. Even the TXT record fetched by processing an exp modifier may be populated with macros. Macros are identified using the percent sign (%). The following macros have been defined

- `%{s}` = the sender's email address
- `%{l}` = the local part of the sender's email address
- `%{o}` = the domain of the sender's email address
- `%{d}` = the current domain, usually the same as the sender's domain but may also have been processed, for example, via the include mechanism
- `%{i}` = the source IP address of the message sender
- `%{p}` = the validated domain name via PTR lookup of the source IP address of the message sender
- `%{v}` = the literal string "in-addr" if the source IP address is an IPv4 address and "ip6" if the source IP address is IPv6
- `%{h}` = the domain part of the HELO/EHLO identity
- `%%` = the literal %
- `%_` = space " "
- `%-` = a URL-encoded space, for example "%20"

The following macros are available for use in the TXT record referenced by an exp mechanism and may not be used elsewhere:

- `%{c}` = the SMTP client IP address
- `%{r}` = the domain name of the host performing the SPF check
- `%{t}` = the current time stamp.

Macro transformers enable use of a subset of the results of a macro, for example, by specifying an integer quantity of domain name labels, or the reversal of the results of a macro, for example, reversing an IP address. Reversal is performed by adding an `r` into the macro curly brackets.

Macro Examples. Consider the example of Figure 10.3, where Mike (mike@ipamww.com) sends me an email to tim@ipamworldwide.com from my SMTP host on IP

addresses 192.0.2.32. Using this and other information from the figure, we can define the macro values for this email transmission as

- %{s} = mike@ipamww.com
- %{l} = mike
- %{o} = ipamww.com
- %{d} = ipamww.com
- %{d3} = ipamww.com
- %{d2} = ipamww.com
- %{d1} = com
- %{i} = 192.0.2.32
- %{ir} = 32.2.0.192
- %{v} = in-addr
- %{h} = ipamww.com
- %{ir}.%{v}._spf.%{d} = 32.2.0.192.in-addr._spf.ipamww.com

SPF provides a powerful macro language to granularly articulate email policies for your organization. However, it is an experimental protocol, as is a close cousin, Sender ID.

10.3.4 Sender ID

Another experimental mechanism for identifying potential spam email is called Sender ID. The Sender ID algorithm seeks to identify whether a given email from a given SMTP client at the given source IP address is authorized to send the email. Like SPF, Sender ID can examine the sender, sender domain, and source IP address of the email message based on the MAIL FROM field. Unlike SPF, Sender ID can also or alternatively verify the sender and sender domain based on message header information. Sender ID, like SPF, utilizes the SPF resource record type, as defined in the previous section with a few modifications:

- the version string ("v = spf1") is replaced with "spf2.0";
- Sender ID includes a scope for the record: "mfrom" indicates the mailfrom entity as in SPF, and/or "pra" the *purported responsible address*, discussed next; and
- modifiers are extended from the SPF definition to enable positional context as an alternative to the SPF-defined global context. That is, a modifier can affect a preceding mechanism, unlike SPF where a modifier is always applied globally.

The scope field is used to derive the sender and sender domain for validation (i.e., the MAIL FROM entity and/or the PRA). The purported responsible address, PRA, scope relates to the identity of the sender closest to the receiving email system. The PRA

algorithm examines the message header, not the envelope, and seeks a sender address by examining the following headers in order, taking the first address found:

- Resent–Sender header
- Resent–From header
- Sender header
- From header

A single valid sender mailbox address (i.e., of the form mailbox@maildomain) found in one of these headers is the PRA. In the simple cases illustrated in Figures 10.3 and 10.4, the purported responsible address would be mike@ipamww.com as derived from the "From" header value. In the case where a third party is used to transmit email on behalf of a legitimate sender, the "Resent–From" or other header value would be used. The term "purported" is used since the algorithm relies on information supplied in the message header, which is supplied by the sender.

There is a bit of controversy around Sender ID versus SPF, which is one of the reasons why both techniques are deemed experimental. For example, Sender ID processing of "v = spf1" (SPF) records could result in valid messages being deemed spam. Hopefully, an agreeable unified approach can be derived in the future.

10.3.5 Domain Keys Identified Mail (DKIM)

DKIM specifies a means for a sender of email to cryptographically sign an email message such that recipients may validate it upon receipt via retrieval and application of the sender's domain key. DKIM utilizes digital signatures, which enable the originator of a given set of data (an email message in this case) to sign the data such that those receiving the data and the signature, along with a corresponding public key for deciphering the signature, can perform data origin and integrity verification. DKIM employs an asymmetric key pair (private key/public key) model. In such a model, the email message and selected header fields are encrypted with a private key and can be validated by decrypting the data with the corresponding public key. The private key and public key form a key pair. The mathematical details are very complex but conceptually, the private/public key pairs provide a means for holders of the public key to verify that data was signed using the corresponding private key. This provides authentication that the data verified was indeed signed by the holder of the private key. Digital signatures also enable verification that the data received matches the data published and was not tampered with in transit.

Referring to Figure 10.5, the data originator, shown on the left of the figure, generates a private key/public key pair and utilizes the private key to sign the data. The first step in signing the data is to produce a hash of the data, sometimes also referred to as a digest. Hashes are one-way functions* to scramble data into a fixed length string for simpler manipulation, and represent a "fingerprint" of the data. This means that it is very

* A one-way function means that the original data is not uniquely derivable from the hash. One can apply an algorithm to create the hash, but there is no inverse algorithm to perform on the hash to arrive at the original data.

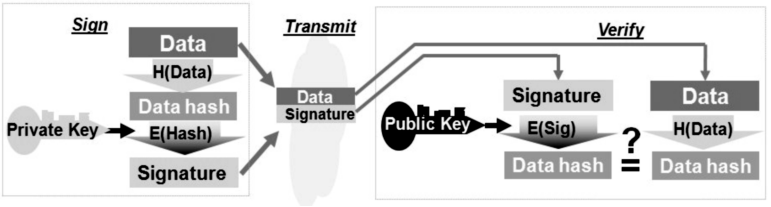


Figure 10.5. Digital signature creation and verification process.

unlikely that another data input could produce the same hash value. Thus, hashes are often used as checksums but don't provide any origin authentication (anyone knowing the hash algorithm can simply hash arbitrary data). Common hash algorithms include HMAC-MD5, RSA-SHA-1, and RSA-SHA-256. DKIM not only uses RSA-SHA-256 by default but also supports RSA-SHA-1. The hash is encrypted using the private key to produce the signature. The encryption algorithm is fed the hash and the private key to produce the signature.

Both the message and its associated signature are transmitted to the recipient. A new email header field, dkim-signature, has been defined to store the DKIM signature with information on retrieving the public key. Based on our prior review of how SMTP works, you may be wondering how modification of envelope data and insertion of headers affect the signature. DKIM offers a "simple" or strict form of canonicalization and a "relaxed" form. The simple form tolerates very little modification while the relaxed form permits white space replacement and header line rewrapping without impacting the signature validity.

DKIM Signature Email Header Field. The recipient must extract the signature from the `dkim-signature` header field. The dkim-signature field also contains the following:

- the DKIM version (e.g., `v=1`)
- the algorithm used to generate the signature (e.g., `a=rsa-sha256`)
- signature (e.g., `b=dqdVx0fAK9...`)
- hash of the canonicalized message body (`bh=7Dkw0eE35Jlkjexcmpol...`)
- canonicalization method (`c=relaxed`)
- the signing domain identifier—the domain of the signing entity (e.g., `d=ipam-worldwide.com`)
- user or agent on whose behalf the message is signed (`i=rooney@ipam-worldwide.com`)
- the selector or key reference within the domain (allows multiple keys per domain which aids in key rollover and more granular signatures) (e.g., `s=europe`)
- enumeration of the header fields that were signed (e.g., `h=from:to:subject:date`)

- additional optional information, including query methods to use to retrieve the public key. The default (and currently only) query method, `q = dns/txt`, instructs the recipient to perform a DNS query of querytype "txt" to retrieve the public key that corresponds with the private key that was used to sign the message. Another optional field of interest, the `i =` tag provides the identity of the user or agent on whose behalf this message was signed.

DKIM TXT Record. Using the query method q = dns/txt, the recipient performs a DNS query for a TXT record within the signing domain. The Question section of the query is formulated by concatenating the selector value (`s =` value), the string "`_domainkey`" and the specified signing domain (`d =` value). Using the example where `s = europe` and `d = ipamworldwide.com` as specified in the `dkim-signature` field of an incoming email, a TXT query for `europe._domainkey.ipamworldwide.com` would be issued. The RData portion of the corresponding TXT record includes one or more tags similar to the dkim-signature field

- DKIM version (`v = DKIM1`)
- Granularity of the key, which if specified, must match the local part of the user or agent (i=) flag in the dkim-signature header (`g = *`)
- Hash algorithm(s) accepted (e.g., `h = sha256`)
- Key type (`k = rsa`)
- Notes for human consumption (`n = updated_key`)
- The public key (`p = Dkjeijf8d98Kz...`)
- Service type (`s = email`)
- Flags indicating such things as the compliance rules among the `i =` tag in the dkim-signature header and the `d =` domain tag (encoded in the TXT record as `t = s`), as well as whether this domain is testing DKIM (`t = y`).

The only required tag is the p tag, the public key. An example TXT record follows

```
europe._domainkey.ipamworldwide.com IN TXT
(''v=DKIM1; p=Dkjeijf98Kz...'')
```

Upon retrieving the public key, the recipient computes a hash of the received message body and signed header fields, as did the originator. The recipient applies the hash algorithm to the received signature using the originator's public key. The output of this decryption, the original data hash, is compared with the recipient's computed hash of the data. If they match, the data has not been modified and the private key holder signed the data.

If an incoming email message contains a dkim-signature header field, it's clear that the sender is using DKIM and has signed the message. But if an incoming email message does not contain a dkim-signature header field, does this mean that the sender does not sign messages? This in fact could create an opening for a SPAM attacker issuing

unsigned email messages from a spoofed source domain. DKIM relies on publication of Author Domain Signing Practices (ADSP), which enables a recipient email server to determine whether the message from a given domain by policy should be signed and if so, by whom and with what signature(s).

A recipient determines the sending domain's signing practices by issuing a query for Qtype = TXT and Qname = `_adsp._domainkey.`*`signing-domain-identifier`*, where *`signing-domain-identifier`* is again the `d` = value. The corresponding TXT record indicates whether email from this domain is always signed, may be signed, and is always signed and any unsigned email should be discarded. Please refer to RFC 5617 (113) for details.

10.3.6 Historic Email Resource Record Types

These resource record types were defined in the early days of DNS and are no longer used. We list them here purely for historical significance.

MR—Mail Rename Record. The MR resource record type translates email to an alias or list into an individual (or multiple, one per MR record). In the simplest sense, it provides an alias for a mailbox name.

Owner	TTL	Class	Type	RData
Emailbox alias name	TTL	IN	MR	Emailbox name
cfo	86400	IN	MR	finance

MB—Mailbox Record. The MB record is defined in RFC 1035 and enables association of a user ID with the desired host containing the user's email box.

Owner	TTL	Class	Type	RData
Email ID	TTL	IN	MB	Mailbox hostname
joe	86400	IN	MB	smtp.ipamworldwide.com.

MG—Mail Group Member Record. RFC 1035 defined the MG resource record to enable association of email users with a user group.

Owner	TTL	Class	Type	RData
Email group name	TTL	IN	MG	Email ID
finance	86400	IN	MG	joe

MINFO—Mailbox/Mailing List Information. The MINFO record was also defined in RFC 1035 and was intended to provide mailbox and mailing list information. It

provides two email box addresses one to request addition to the mailing list and another to report errors.

Owner	TTL	Class	Type	RData	
Mailbox name	TTL	IN	MINFO	Requests mailbox	Errors mailbox
newsalerts	86400	IN	MINFO	hostmaster	majordomo

10.4 SECURITY APPLICATIONS

10.4.1 Securing Name Resolution—DNSSEC Resource Record Types

Chapter 13 is devoted to the topic of DNS security extensions (DNSSEC), so we will summarize the DNSSEC required resource record types for completeness within this chapter. We'll provide the full context and description of DNSSEC in Chapter 13.

DNSKEY—DNS Key Record. The DNSKEY resource record is used in DNSSEC to publish public keys used for validating signatures on zone information. The server signs its authoritative resource record sets within a zone using a private key and the corresponding public key is published in the zone file in the form of the DNSKEY record. Two types of keys are published: a zone signing key (ZSK), which signs resource record data and a key signing key (KSK) which signs the ZSK. The resolver can use this public key to validate a given RRSet's signature.

Owner	TTL	Class	Type	RData			
Key name	TTL	IN	DNSKEY	Flags	Protocol	Algorithm	Key
ipamww.com.	86400	IN	DNSKEY	256	3	5	AweE8F(le...

In this example, the RData fields are interpreted as follows:

- the Flags field provides information on the type and status of the key. Currently defined values for the Flags field are as follows.
  - Bit 7. This key is a zone signing key (Decimal = 256)
  - Bit 8. Revoke this Key
  - Bit 15. This key is a key signing key (Decimal = 1)
  - Other bits. Unassigned
- the Protocol field must have a value of "3" indicating DNSSEC (this is the only value currently defined).
- the Algorithm has a value of "5" in the example above, indicating the RSA-SHA-1 algorithm. Algorithms currently supported are encoded as follows:
  - Value = 1. RSA/MD5, which is not recommended according to RFC 4034

- Value = 2. Diffie–Hellman
- Value = 3. DSA*/SHA-1
- Value = 4. Reserved for Elliptic Curve
- Value = 5. RSA-SHA-1, which is mandatory according to RFC 4034
- Value = 6. DSA-NSEC3-SHA-1—an alias for algorithm 3, but with the qualifier that NSEC3 records instead of NSEC records are used
- Value = 7. RSA-SHA-1-NSEC3-SHA-1—an alias for algorithm 5, but with the qualifier that NSEC3 records instead of NSEC records are used
- Value = 8. RSA-SHA-256
- Value = 10. RSA-SHA-256
- Value = 12. GOST R 34.10-2001
- Value = 252. Indirect
- Values = 253-254. Private
- Values = 0, 123-251, 255. Reserved
- Other values. Unassigned.

- the Key is the public key.

DS—Delegation Signer Record. RFC 4034 (114) defines the DS resource record type, which essentially extends the chain of trust to a signed delegated domain (zone). The DS resource record enables a parent zone to authenticate its child zone's public Key Signing Keys (DNSKEY record for the KSK). As such, each DS record refers to a specific (by key tag) DNSKEY resource record in the delegated child zone. Authenticating the DS record enables clients to authenticate the child zone's DNSKEY.

Owner	TTL	Class	Type	RData			
Delegated domain	TTL	IN	DS	Key tag	Alg.	Type	Digest
child.ipamww.com.	86400	IN	DS	32284	5	1	75CF28D3OQ35...

The Algorithm field identifies the algorithm field on the corresponding DNSKEY record. The DS record refers to a DNSKEY record by including a digest (hash) of the DNSKEY RR in the Digest field; the [Digest] Type field indicates the algorithm utilized to construct the digest.

DLV—DNSSEC Lookaside Validation Record. Specified in RFC 4431 (115), the DLV resource record is used within DNSSEC for publishing trust anchors outside of the normal DNS domain tree hierarchy, that is, the chain of trust. The DLV record is structured identically to the DS record in that it identifies a "proxy parent zone" and it thus authenticates the "child" zone's public key signing key records (DNSKEY).

* DSA = U.S. Government Digital Signature Algorithm.

Lookaside validation is intended to provide an alternative upstream trust anchor, such as dlv.isc.org, in the absence of root and TLD zone signings.

Owner	TTL	Class	Type	RData			
DLV domain	TTL	IN	DLV	Key tag	Alg.	Type	Digest
ipamww.com.dlv_reg.net.	86400	IN	DLV	32284	5	1	90df80DF89lLe...

NSEC—Next Secure Record. The NSEC resource record type provides two sets of information. The set of NSEC RRs in a zone forms a chain of authoritative owner names in the zone and indicates which authoritative RRSets exist in the zone. The NSEC resource record contains the next owner name that identifies associated authoritative owner names within the chain, as well as the set of RR types present at the NSEC resource record's owner name.

Owner	TTL	Class	Type	RData	
RRSet Owner	TTL	IN	NSEC	Next RRSet Owner	Type Bit Maps
ns1.ipamww.com.	86400	IN	NSEC	ns2.ipamww.com.	A NS RRSIG NSEC

The Next RRSet Owner field contains the next owner name in the canonical ordering of the zone that has authoritative data or contains an RRSet of type NS defining a delegation point. This provides authenticated denial of existence of resource records between the RRSet identified within the NSEC Owner field and the Next RRSet Owner RData field. The Type Bit Maps field identifies the resource record types that exist at this NSEC resource record's owner name. Within this field, if a bit = 1, then the RRType corresponding to this bit number exists. Thus if bit 1 is 1, corresponding to RR Type = 1 or A record, then an A RRSet is present. Fortunately, the text representation of this is in the familiar resource record type mnemonic.

NSEC3—NSEC3 Record. The NSEC resource record provides authenticated denial of existence for RRSets, but it also enables easy enumeration of RRSets in the zone, which can be considered an information security risk. In other words, a curious or malicious querier could attempt to resolve a bogus name and receive the pair of resource record owner names surrounding the queried hostname.

Like NSEC, the NSEC3 record provides authenticated RRSet denial of existence, but obfuscates the chain of RRSets in the zone. This obfuscation renders the footprinting of a zone's contents much more computationally intensive. Instead of pointing to the new owner name field, NSEC3 points to the next hashed owner name field in hash order. And the salt value that is appended to each owner name prior to hash generation further complicates the generation of hashed owner names by someone attempting to footprint the zone.

For each RRSet in the zone, the owner field is hashed using the specified hash algorithm applied to the owner name concatenated with the salt field iteratively <Iterations> + 1 times. The following pseudocode states this in another way:

```
x = {RRSet owner field concatenated with Salt value}
y = H (x)        a hash of x as defined in the prior statement
for (i = Iterations value; i > 0; i-) {
     y = H (y)
}
```

Owner	TTL	Class	Type	RData							
Hashed RRSet Owner	TTL	IN	NSEC3	Hash Alg.	Flags	Iterations	Salt Len.	Salt	Hash Len.	Next Hashed Owner Name	Type Bit Maps
jAdfJE; . . .	8640	IN	NSEC3	1	0	2	8	a808f6ce 1a950b1c	18	k0Lse7. . .	A RRSIG NSEC3

The RData fields for the NSEC3 record are defined as follows:

- *Hash Algorithm*. The algorithm used to construct the hash value; valid values are
  - Reserved
  - 1 = RSA-SHA-1
  - 2-255. Unassigned
- *Flags*. Consisting of a set of eight boolean flags, the Flags field has currently a single flag defined (bit 0). If bit 0 is set, this indicates that this record covers one or more unsigned delegation records. This Opt-Out flag enables "opting out" of securing delegations to unsigned zones (i.e., validating the non-existence of a child zone's DS record).
- *Iterations*. Specifies the number of additional applications of the hash function.
- *Salt Length*. Included in the wire format but not presented in the resource record text format, this field indicates the length in bytes of the Salt field (valid values = 0–255).
- *Salt*. The value of the Salt field is appended to the RRSet owner prior to application of the hash function and is represented in case-insensitive hexadecimal.
- *Hash Length*. The length in octets of the next hashed owner name field, included on the wire but not represented in resource record text format.
- *Next Hashed Owner Name*.
- *Type Bit Maps*. This field defines the resource record types defined for this owner within the zone and is encoded in the same manner as the corresponding field in the NSEC record.

NSEC3PARAM—NSEC3 Parameters Record. The NSEC3PARAM record type defines the parameters needed to compute hashed owner names and hence the corresponding NSEC3 records within the zone upon signing. The NSEC3PARAM record is used by the server to identify negative answers in response to a query. Thus, when a query arrives for a nonexistent RRSet within the zone, the server applies the NSEC3PARAM parameters to hash the queried owner name in order to provide an appropriate NSEC3 response, that is, between which two hashed RRSets does this queried owner name fall? Only one NSEC3PARAM record should be present within the zone file. The NSEC3PARAM record is also used by the server when signing new or changed RRSets automatically.

The RData fields have identical meanings as corresponding fields within the NSEC3 RData fields.

Owner	TTL	Class	Type	RData				
Domain name	TTL	IN	NSEC3PARAM	Hash Alg.	Flags	Iterations	Salt Len.	Salt
ipamww.com.	86400	IN	NSEC3PARAM	1	0	2	8	a808f6ce1a950b1c

RRSIG—Resource Record Set Signature Record. The Resource Record Set Signature resource record contains the digital signature associated with a given RRSet. This signature, along with the zone's public [zone signing] key, are used to authenticate the corresponding RRSet's integrity and origin.

Owner	TTL	Class	Type	RData							
RRSet Owner	TTL	IN	RRSIG	Type Cov.	Alg.	Labels	Orig. TTL	Expire	Inception	Key tag	Signer	Signature
ftp1.ipamww.com.	86400	IN	RRSIG	A	5	3	86400	20080515133509	20080115133509	27783	ipamww.com.	N78E...

The RData fields within the RRSIG record are defined as follows:

- *Type Covered.* The resource record type of the corresponding owner and class signed by this signature. This field is the standard resource record type discussed for resource records throughout this chapter. In the example above, the A (address) resource record type indicates that A records with name = ftp1.ipamww.com (Owner field) of class IN are signed with this RRSIG record.
- *Algorithm.* The algorithm used in generating the data hash for comparison with the received signature. This field is encoded in the same manner as the Algorithm field of the DNSKEY resource record type.
- *Labels.* Indicates the number labels. Recall that labels refer to the text representation of domain names, with a label for each name "between the dots." Thus, www.ipamworldwide.com has three labels. This field is used to reconstruct the

original owner name used to create the signature in the case where the owner name returned by the server has a wildcard label (*).

- *Original TTL*. The TTL of the signed RRSet as defined in the authoritative zone, used to validate a signature. This field is needed because the TTL field returned in the original response is normally decremented by a caching resolver and use of that TTL value may lead to erroneous calculations.
- *Signature Expiration*. The date and time of the expiration of this signature expressed as either the number of seconds since January 1, 1970 00:00:00 UTC or in the form of YYYYMMDDHHmmSS where
  - YYYY is the year
  - MM is the month, 01–12
  - DD is the day of the month, 01–31
  - HH is the hour in 24 h notation, 00–23
  - mm is the minute, 00–59
  - SS is the second, 00–59
  - Signatures are not valid after this date/time.
- *Signature Inception*. The date and time of the inception of this signature formatted in the same manner as the Signature Expiration field. Signatures are not valid before this date/time.
- *Key Tag*. Provides an association with the corresponding DNSKEY resource record that can be used to validate the signature.
- *Signer's Name*. Identifies the owner name of the DNSKEY resource record (i.e., the domain name) that is to be used to validate this signature.
- *Signature*. The cryptographic signature covering the resource record set defined by this RRSIG owner, class, and covered type fields and this RRSIG RData fields (excluding this Signature field).

10.4.2 Other Security-Oriented DNS Resource Record Types

TA—Trust Authority Record. While an RFC does not exist defining the TA resource record, IANA has assigned it a value, so we'll mention it here. The TA resource record is identical in format to the DS record type including RData fields for key tag, algorithm, digest type, and digest. Use of the TA record enables a resolver to have a resource record signature validated by a known trust authority even if the root zone has not been signed (it has now been signed!). This functionality is now provided using the DLV record.

CERT—Certificate Record. RFC 4398 (116) defines the CERT record as a means to store certificates and certificate revocation lists (CRLs) in DNS. Certificates provide a means to identify an organization, server, individual, or other entity and associate a public key with that identity. The public key can be used to authenticate the sender's identity and to encrypt and decrypt communications and validate message integrity.

Certificates are hierarchical and can be used to validate up to a known trusted entity (Certificate Authority). CRLs are lists of certificates, which have been revoked due to expiration or manual revocation.

CERT records containing certificates are stored in DNS to enable resolvers to obtain certificates via DNS instead of from a destination certificate server. The CERT resource record has the following format.

Owner	TTL	Class	Type	RData			
Domain name	TTL	IN	CERT	Certificate Type	Key tag	Algorithm	Certificate or CRL
ipamww.com.	86400	IN	CERT	PGP	436	3	A4df480DFC9lLa. . .

The owner field identifies the entity to which the certificate applies when a certificate is included in the RData portion of the record. If a CRL is included in the RData section, the owner name should contain the domain name related to the issuing authority. The RData portion contains the following subfields

- Certificate Type such as X.509/PKIX, PGP, and others
- Key tag, which is used to streamline the identification of relevant certificates to those of matching key tags
- Algorithm. The algorithm used in generating the key, which is encoded in the same manner as the Algorithm field of the DNSKEY resource record type.
- The certificate or CRL

IPSECKEY—Public Key for IPSec Record. The IPSECKEY resource record type, defined in RFC 4025 (117), provides a means to store a public key in DNS for use with IPSEC. This resource record enables a client seeking to establish an IPSec tunnel to a remote host to identify a means to authenticate the remote host and to determine whether to connect directly to the host or connect via another node acting as a gateway. IPSECKEY resource records are associated with the intended remote host's IP address or host domain name. IP addresses are stored in the .arpa. reverse domain space. The format of the IPSECKEY resource record is as follows.

Owner	TTL	Class	Type	RData				
IP address in .arpa. domain or host domain name	TTL	IN	IPSECKEY	Precedence	Gateway Type	Algorithm	Gateway	Public key
1.0.12.10. in-addr.arpa.	86400	IN	IPSECKEY	10	1	2	10.100.1.2	Adf4C9lL. . .

The RData field contains the following fields:

- *Precedence*. Used to prioritize multiple records within a common RRSet, using the lowest precedence first.
- *Gateway Type*. Indicates the format of the Gateway field.
  - 0 = no gateway is present
  - 1 = IPv4 address
  - 2 = IPv6 address
  - 3 = FQDN
- *Algorithm*. The format of the Public Key field.
  - 0 = no key is present
  - 1 = DSA formatted key
  - 2 = RSA formatted key
- *Gateway*. Identifies a gateway to which an IPSec tunnel can be established to reach the remote host (identified by the owner field). The interpretation of this field is governed by the Gateway Type field.
- *Public Key*. The key generated using the algorithm specified in the Algorithm field.

KEY—Key Record. The KEY record was defined with the initial incarnation of DNSSEC, but was superseded by the DNSKEY resource record. However, prior to the release of DNSSEC*bis*, the KEY record was also utilized to store public keys associated with the SIG(0) record. The KEY record has the same format as the DNSKEY record.

Owner	TTL	Class	Type	RData			
Key name	TTL	IN	KEY	Flags	Protocol	Algorithm	Key
K3941.ipamww.com.	86400	IN	KEY	256	3	1	12S9X-weE8F(le...

KX—Key Exchanger Record. The KX record enables specification of an intermediary that can supply a key on behalf of another host. In other words, if intending to perform key negotiation with x.ipamworldwide.com, the KX record could point to the y.ipamworldwide.com host domain name with whom key exchange negotiation should ensue. A preference field enables specification of multiple alternate domains of varying preference for key negotiation.

Owner	TTL	Class	Type	RData	
Host domain name	TTL	IN	KX	Preference	Key exchanger host domain name
x.ipamworldwide.com.	86400	IN	KX	10	y.ipamworldwide.com.
x.ipamworldwide.com.	86400	IN	KX	20	z.ipamworldwide.com.

SIG—Signature Record. The SIG resource record has been superseded by the RRSIG record within the scope of DNSSEC, though the SIG record is still in use for digitally signing DNS updates and zone transfers outside the scope of DNSSEC. That is, you don't need to deploy DNSSEC to enable transaction signatures of updates and zone transfers. Such transactions can be signed using shared secret keys via TSIG (Transaction Signature) records or by using private/public key pairs via SIG(0), where corresponding public keys are stored as KEY records. The notation SIG(0) refers to the use of the SIG resource record with an empty (0) Type Covered field. In such cases, RFC 2931 (118) recommends setting the owner field to root, the TTL to 0, and class to ANY as shown in the example below.

The SIG record is formatted identically to the RRSIG record, with the exception of the formatting of the Expiration Date and Inception Date fields; for the SIG record, these fields are not formatted by date per the RRSIG record and are instead formatted as an incremental integer, enumerated as the number of seconds since January 1, 1970 00:00:00 UTC. This counter will rollover to 0 and continue counting after the counter exceeds 4.29 billion seconds (a little over 136 years).

Owner	TTL	Class	Type	RData							
RRSet Domain	TTL	IN	SIG	Type Cov.	Alg.	Labels	Orig. TTL	Expire	Inception	Key tag	Signer	Signature
.	0	ANY	SIG	0	3	3	86400	2008051 5133509	2008011 5133509	26421	ipamw w.com.	Zx9v...

SSHFP—Secure Shell Fingerprint Record. The Secure Shell (SSH) Protocol enables secure login from a client to a server and other secure network services over an insecure IP network. The security of the connection relies upon the user authenticating him- or herself to the server as well as the server authenticating itself to the client via Diffie–Hellman key exchange. If the public key is not already known by the client, a fingerprint of the key is provided by the server for verification by the user. Storage of this key fingerprint in DNS provides a means for the client to lookup and verify the fingerprint out of band via a "third party." The lookup requires use of DNSSEC to secure the lookup process and assure message integrity. The SSHFP resource record is the record type used to store these SSH fingerprints.

Owner	TTL	Class	Type	RData		
Host domain name	TTL	IN	SSHFP	Algorithm	Fingerprint type	Fingerprint
srv21.ipamww.com.	86400	IN	SSHFP	2	1	8Fd7q90D + fd...

The RData portion of the SSHFP record includes the following fields:

- *Algorithm.* Currently defined values are
  - 0 = Reserved

- 1 = RSA
- 2 = DSA

- *Fingerprint Type*. Currently defined values are
  - 0 = Reserved
  - 1 = SHA-1
- Key Fingerprint

10.4.3 Geographical Location Lookup

GPOS—Geographical Position Record. The GPOS resource record type, originally defined in RFC 1712 (119), has been superseded by the LOC resource record type. GPOS encoded the longitude, latitude, and altitude of a host as shown below.

Owner	TTL	Class	Type		RData	
Host domain name	TTL	IN	GPOS	Longitude	Latitude	Altitude
srv1.ipamww.com.	86400	IN	GPOS	39.582	-75.801	128.2

LOC—Location Resource Record. This type of resource record enables encoding of latitude, longitude, and altitude information about the respective host. RFC 1876 (120) defines the LOC record, which obsoletes the GPOS resource record type. The RData field for the LOC record presents each coordinate in the three dimensions

- *Latitude*. Degrees [minutes [seconds]] "N" or "S"
- *Longitude*. Degrees [minutes [seconds]] "E" or "W"
- *Altitude*. Altitude in meters
- Precision of each measure as diameter of "sphere of error" in meters

Owner	TTL	Class	Type	RData			
Host domain name	TTL	IN	LOC	Latitude	Longitude	Altitude	Precision
srv-97.ipamww.com.	86400	IN	LOC	39 58 N	75 38 W	128	50 m

In the example above, the hostnamed srv-97.ipamww.com is located at 39°58′ N latitude, 75°38′ W longitude, is 128 m above sea level, all within a sphere of error with diameter 50 m.

10.4.4 Non-IP Host-Address Lookups

ISDN—Integrated Services Digital Network Record (Experimental). The ISDN type enables association of an ISDN address to a host. The ISDN address is

the form of a telephone number, as defined by the International Telecommunications Union standard E.164. The subaddress field is optional.

Owner	TTL	Class	Type	RData	
Host domain name	TTL	IN	ISDN	ISDN Address	Subaddress
isdnhost.ipamww.com.	86400	IN	ISDN	16105551298	318

NSAP—Network Service Access Point Record. The NSAP resource record enables translation of a hostname or FQDN to a Network Service Access Point (NSAP) address. NSAP is the notation for a network device that supports the ISO Connectionless Network Protocol (CLNP). Without getting into the details of NSAP addresses, which never really caught on, the NSAP resource record functions equivalently to an A record for IPv4 and AAAA for IPv6. It provides a destination address for a queried hostname.

Owner	TTL	Class	Type	RData
Host domain name	TTL	IN	NSAP	NSAP Address
nsap-host.ipamww.com.	86400	IN	NSAP	47.0005.09.d78d01.1010.0ffe.0011. . .00

NSAP-PTR—Network Service Access Point Reverse Record. The NSAP-PTR record type performs the equivalent pointer record functionality for NSAP addresses, linking an NSAP address suffix to a host domain name. The nsap.int domain serves as the corresponding reverse TLD. As with IP address based pointer records, the NSAP address must be reversed, and dots inserted between each digit. Finally, the nsap.int. suffix is added.

Owner	TTL	Class	Type	RData
NSAP Address Reversed	TTL	IN	NSAP-PTR	Host domain name
0.0. . .1.1.0.0.e.f.f.0.0.1.0.1.1.0.d.8.7.d.9.0.5.0.0.0.7.4.nsap.int.	86400	IN	NSAP-PTR	nsap-host.ipamww.com.

PX—Pointer for X.400. The PX resource record is defined in RFC 2163 (121) and is intended to provide a mapping between DNS domain names and an X.400 address for email address mapping. X.400 is an OSI standard for messaging or email, though today most systems use the Simple Mail Transfer Protocol. This resource record type is useful for networks containing SMTP-to-x.400 email gateways, referred to as MIXER (MIME Internet X.400 Enhanced Relay) gateways. The X.400 address is formatted using the Originator/Recipient (O/R) convention.

Owner	TTL	Class	Type	RData		
Domain name	TTL	IN	PX	Preference	DNS domain	X.400 mapping
ipamww.com.	86400	IN	PX	10	ipamww.com.	O = company.PRMD-netx.ADMD.C = tv.

X25—X.25 PSDN Address Record (Experimental). This is an experimental resource record and is not widely used, as X.25 packet switched data networks (PSDNs) are not widely in-use today. It has a number of possible applications

- document the addresses to use in static configurations of IP-to-X.25 and SMTP-to-X.25;
- automatically associate an IP address to PSDN address; and
- configure names to X.25 PSDN addresses.

It also provides a function similar to ARP for wide area nonbroadcast networks.

Owner	TTL	Class	Type	RData
Host domain name	TTL	IN	X25	PSDN Address
x25-host.ipamww.com.	86400	IN	X25	31161700956

RT—Route Through. The Route Through resource record was defined in RFC 1183 (108) and is used to denote a proxy or alternative destination to which to route traffic for hosts without a direct network link. Multiple route through hosts can be identified, each with associated preference values, much like the MX resource record.

Owner	TTL	Class	Type	RData	
Host domain name	TTL	IN	RT	Preference	Proxy Hostname
host.ipamww.com.	86400	IN	RT	10	proxy.ipamww.com.

10.4.5 The Null Record Type

NULL. The NULL resource record type is experimental and enables specification of up to 65,535 bytes of "anything." It is usually ignored and not widely used.

Owner	TTL	Class	Type	RData
Host domain name	TTL	IN	NULL	Up to 65,535 bytes of "anything"
host.ipamww.com.	86400	IN	NULL	"Ignore this NULL resource record!"

10.5 EXPERIMENTAL NAME–ADDRESS LOOKUP RECORDS

10.5.1 IPv6 Address Chaining—The A6 Record (Experimental)

Given the sheer length of IPv6 addresses, the IETF had considered an iterative approach to resolving hostnames to IPv6 addresses. The A6 record, defined in RFC 2874 (122), intended to map a host domain name to a portion (or all) of an IPv6 address, with pointers for the resolver to iteratively resolve the remainder of the IPv6 address to its full 128 bits. This enabled resolution of the host domain name by starting most commonly with the interface ID, then adding in the appropriate subnet ID, and global routing prefix, essentially resolving the hostname address moving from right to left. The intent was to simplify renumbering of IPv6 networks that may be necessary due to network maintenance, changing of ISPs, or other reasons. Changing the subnet ID for a number of hosts was as simple as changing one record, instead of each host's record.

However, due to the complexity in accurately configuring DNS with the appropriate linkages (and preventing open linkages), this resource record type was changed to experimental status. To illustrate this, the example below illustrates the A6 resource record and how three successive queries would be used to fully resolve. Note that more or fewer linkages could be defined based on individual preference.

Owner	TTL	Class	Type	RData		
Host domain name	TTL	IN	A6	Prefix Length	Address Suffix	Prefix Name
ftp-sf.ipamww.com.	86400	IN	A6	64	::A05F:0:0:2001	sf-net.ipamww.com.
sf-net.ipamww.com.	86400	IN	A6	48	0:0:0:8400::	na-west.ipamwwe.com.
na-west.ipamww.com.	86400	IN	A6	0	2001:DB8:4AF0::	

Note that the RData portion of the A6 resource record contains three subfields. The prefix length indicates the number of offset bits from the start of the address to begin inserting the address suffix bits. Thus, the first listed A6 record with owner field "ftp-sf.ipamww.com." indicates a prefix length of 64 bits, specifying the interface identifier of ::A05F:0:0:2001.

The prefix name field provides a linkage to a second lookup to continue building the entire 128-bit address. In this case, we are linked to the "sf-net.ipamww.com." prefix name, which points to an A6 record with owner field, "sf-net.ipamww.com." The corresponding A6 record indicates a 48-bit prefix length with IPv6 address, 0:0:0:8400::. Note that the full IPv6 address notation is used, including the restriction of a single double colon. This record then points to the na-west.ipamww.com. A6 record, which completes our formulation of the IPv6 address for resolution with its zero offset. Figure 10.6 illustrates this process.

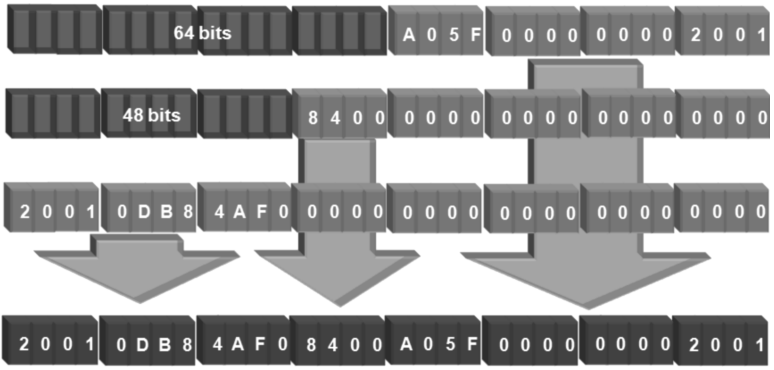


Figure 10.6. Iterative derivation of an IPv6 address using A6 records.

10.5.2 APL—Address Prefix List Record (Experimental)

While A and AAAA records are used to resolve host IP addresses, the APL record seeks to resolve address prefixes or subnet addresses. The following example illustrates a scenario of advertising a set of address ranges associated with a domain or host. The RData portion of the APL record consists of an optional negation character (!), the address family as defined by IANA* followed by a colon, then the address in CIDR notation (network/prefix length).

Owner	TTL	Class	Type	RData
Host domain name	TTL	IN	APL	Address Family:Address/Prefix
sf-ftp.ipamww.com.	86400	IN	APL	1:10.0.128/18, !10.16.128.0/18 2:2001:DB8:4AF0:8400::/56

In the above example, address prefixes associated with sf-ftp.ipamww.com.com are 10.0.128.0/18 for IPv4, not 10.16.128.0/18 for IPv4 and the prefix 2001:DB8:4AF0:8400::/56 for IPv6.

10.6 RESOURCE RECORD SUMMARY

Table 10.1 summarizes the currently defined set of resource records in alphabetical order by resource record type (RRType—also corresponds to valid QType when a querier seeks this type of information from DNS, that is, within the Question section of a DNS message). While not all resource records are IETF standards or even defined within the IETF, most of those that have been assigned an RR Type ID number by IANA are listed here. Current IETF status is provided along with the defining document, which can be accessed for more details.

* Address family values are maintained by IANA, see http://www.iana.org/assignments/address-family-numbers. Relevant to our example, IANA has assigned family number 1 to IPv4 and 2 to IPv6.

TABLE 10.1. Resource Record and Query Type Summary

RRType (or QType)	RR Purpose (i.e., RData Contents)	RR Type ID	IETF Status	Defining Document
A	IPv4 address for a given hostname	1	Standard	RFC 1035 (99)
AAAA	IPv6 address for a given hostname	28	Draft Standard	RFC 3596 (123)
A6	IPv6 address or portion thereof for iterative IPv6 address resolution for a given hostname	38	Experimental	RFC 2874 (122)
AFSDB	Server hostname for a given AFS and DCE domain	18	Experimental	RFC 1183 (108)
APL	Address prefix lists for a given domain	42	Experimental	RFC 3123 (124)
ATMA	Asynchronous Transfer Mode (ATM) address for a host	34	Not Submitted	ATM Name System Specification by the ATM Forum (125)
CERT	Certificate or Certificate Revocation List	37	Standards Track	RFC 4398 (116)
CNAME	Alias hostname for a host	5	Standard	RFC 1035 (96)
DHCID	Associates a DHCP client's identity with a DNS name	49	Standards Track	RFC 4701 (126)
DLV	Authoritative zone signature for a trust anchor	32769	Informational (DNSSEC)	RFC 4431 (115)
DNAME	Alias domain name	39	Proposed Standard	RFC 2672 (107)
DNSKEY	Authoritative zone signature within a chain of trust	48	Standards Track (DNSSEC)	RFC 4034 (114)
DS	Signature for delegated child zone	43	Standards Track (DNSSEC)	RFC 4034 (114)
GID	Group ID	102	RESERVED	IANA-Reserved
GPOS	Lat./long./altitude for a given host - superseded by LOC	27	Experimental	RFC 1712 (119)
HINFO	CPU and OS information for a host	13	Standard	RFC 1035 (99)
HIP	Host Identity Protocol	55	Experimental	RFC 5205 (127)
IPSECKEY	Public key for a given DNS name for use with IPSec	45	Proposed Standard	RFC 4025 (117)

(*continued*)

TABLE 10.1. Resource Record and Query Type Summary (*Continued*)

RRType (or QType)	RR Purpose (i.e., RData Contents)	RR Type ID	IETF Status	Defining Document
ISDN	Integrated Services Digital Network (ISDN) address and subaddress for a given host	20	Experimental	RFC 1183 (108)
KEY	Superseded by DNSKEY within DNSSEC but still used by SIG(0) and TKEY	25	Proposed Standard	RFC 2536 (128)
KX	Intermediary domain to obtain a key for a host in given domain	36	Informational	RFC 2230 (129)
LOC	Lat./long./altitude and precision for a given host	29	Uncommon	RFC 1876 (120)
MB	Mailbox name for a given email ID	7	Experimental	RFC 1035 (99)
MD	Mail delivery host for a given domain	3	Obsolete	RFC 1035 (99)
MF	Host that will accept mail for forwarding to a given domain	4	Obsolete	RFC 1035 (99)
MG	Mail group mailbox name for a given email ID	8	Experimental	RFC 1035 (99)
MINFO	Mailbox names for sending account requests or error reports for a given mailbox name	14	Experimental	RFC 1035 (99)
MR	Alias for a mailbox name	9	Experimental	RFC 1035 (99)
MX	Mail exchanger for email host resolution	15	Standard	RFC 1035 (99)
NAPTR	Uniform resource identifier for a generic string used for DDDS, ENUM applications	35	Standards Track	RFC 3761 (111)
NS	Name server for a given domain name	2	Standard	RFC 1035 (99)
NSAP	Network Services Access Point address for a host	22	Uncommon	RFC 1706 (130)
NSAP-PTR	Hostname for a given NSAP address	23	Uncommon	RFC 1706 (130)
NSEC	Authenticated confirmation or denial of existence of a resource record set for DNSSEC	47	Standards Track (DNSSEC)	RFC 4034 (114)

TABLE 10.1. Resource Record and Query Type Summary (*Continued*)

RRType (or QType)	RR Purpose (i.e., RData Contents)	RR Type ID	IETF Status	Defining Document
NSEC3	Authenticated denial of existence of a resource record set for DNSSEC (without trivial zone enumeration obtainable with NSEC)	50	Standards Track (DNSSEC)	RFC 5155 (131)
NSEC3 PARAM	NSEC3 parameters used to calculate hashed owner names	51	Standards Track (DNSSEC)	RFC 5155 (131)
NULL	Up to 65,535 bytes of anything for a given host	10	Experimental	RFC 1035 (99)
NXT	Superseded by NSEC	30	Obsolete (DNSSEC)	RFC 3755 (132)
PTR	Hostname for a given IPv4 or IPv6 address	12	Standard	RFC 1035 (99)
PX	X.400 mapping for a given domain name	26	Uncommon	RFC 2163 (121)
RP	Email address and TXT record pointer for more info for a host	17	Experimental	RFC 1183 (108)
RRSIG	Signature for a resource record set of a given domain name, class and RR Type	46	Standards Track (DNSSEC)	RFC 4034 (114)
RT	Proxy hostname for a given host that is not always connected	21	Experimental	RFC 1183 (108)
SIG	Superseded by RRSIG within DNSSEC; used by SIG(0) and TKEY	24	Proposed Standard	RFC 2536 (128)
SOA	Authority information for a zone	6	Standard	RFC 1035 (99)
SPF	Sender Policy Framework enables a domain owner to identify hosts authorized to send emails from the domain	99	Experimental	RFCs 4408 (112), 4409 (133)
SRV	Host providing specified services in a domain	33	Standards Track	RFC 2782 (134)
SSHFP	Secure Shell fingerprints enables verification of SSH host keys using DNSSEC	44	Standards Track	RFC 4255 (135)

(*continued*)

TABLE 10.1. Resource Record and Query Type Summary (*Continued*)

RRType (or QType)	RR Purpose (i.e., RData Contents)	RR Type ID	IETF Status	Defining Document
TXT	Arbitrary text associated with a host	16	Standard	RFC 1035 (99)
UID	User ID	101	RESERVED	IANA-Reserved
UINFO	User Info	100	RESERVED	IANA-Reserved
UNSPEC	Unspecified	103	RESERVED	IANA-Reserved
WKS	Services available via a given protocol at a specified IP address for a host SRV RR more commonly used today	11	Standard	RFC 1035 (99)
X25	X.25 PSDN	19	Experimental	RFC 1183 (108)