Network Working Group M. Lentczner
Internet-Draft M. Wong
Expires: April 12, 2005 October 12, 2004

Sender Policy Framework: Authorizing Use of Domains in MAIL FROM
draft-lentczner-spf-00

Status of this Memo

Copyright Notice

Abstract

Mail on the Internet can be forged in a number of ways. In particular, existing protocols place no restriction in what a sending host can use as the reverse-path of a message. This document describes a protocol whereby a domain can explicitly authorize the hosts that are allowed to use its domain name in a reverse-path, and a way for receiving hosts to check such authorization.

Table of Contents

1. Introduction 4
1.1 Protocol Status 4
1.2 Terminology 5
2. Operation 6
2.1 The Mail From Identity 6
2.2 Publishing Authorization 6
2.3 Checking Authorization 6
2.4 Interpreting the Result 7
2.4.1 Neutral 8
2.4.2 Pass 8
2.4.3 Fail 8
2.4.4 SoftFail 8
2.4.5 None 9
2.4.6 TempError 9
2.4.7 PermError 9
3. SPF Records 10
3.1 Publishing 10
3.1.1 RR Types 10
3.1.2 Multiple Records 11
3.1.3 Additional Records 11
3.1.4 Multiple Strings 11
3.1.5 Record Size 12
3.1.6 Wildcard Records 12
4. The check_host() Function 13
4.1 Arguments 13
4.2 Results 13
4.3 Initial Processing 14
4.4 Record Lookup 14
4.5 Selecting Records 14
4.6 Record Evaluation 15
4.6.1 Term Evaluation 15
4.6.2 Mechanisms 16
4.6.3 Modifiers 16
4.7 Default result 17
4.8 Domain Spec 17
5. Mechanism Definitions 18
5.1 "all" 18
5.2 "include" 19
5.3 "a" 20
5.4 "mx" 20
5.5 "ptr" 21
5.6 "ip4" and "ip6" 21
5.7 "exists" 22
6. Modifier Definitions 23
6.1 redirect: Redirected Query 23
6.2 exp: Explanation 24

7. Miscellaneous 26
7.1 Unrecognized Mechanisms and Modifiers 26
7.2 Processing Limits 26
8. Macros 28
8.1 Macro definitions 28
8.2 Expansion Examples 31
9. Implications 32
9.1 Sending Domains 32
9.2 Mailing Lists 32
9.3 Forwarding Services 32
9.4 Mail Services 33
9.5 MTA Relays 33
10. Security Considerations 35
11. IANA Considerations 37
12. Contributors and Acknowledgements 38
13. References 39
13.1 Normative References 39
13.2 Informative References 39
Authors' Addresses 40
A. Collected ABNF 41
B. Extended Examples 43
B.1 Simple Examples 43
B.2 Multiple Domain Example 44
B.3 RBL Style Example 45
Intellectual Property and Copyright Statements 46

1. Introduction

The current e-mail infrastructure has the property that any host injecting mail into the mail system can identify itself as any domain name it wants. Hosts can do this at a variety of levels: in particular, the session, the envelope, and the mail headers. While this feature is desirable in some circumstances, it is a major obstacle to reducing end-user unwanted e-mail (or "spam"). Furthermore, many domain name holders are understandably concerned about the ease with which other entities may make use of their domain names, often with intent to impersonate.

This document defines a protocol by which hosts may be authorized by domains to use the domain name in the envelope "Mail From" identity. Compliant domain name holders publish SPF records about which hosts are permitted to use their names, and compliant mail receivers use the published SPF records to test the authorization of hosts using a given "Mail From" identity during a mail transaction.

An additional benefit to mail receivers is that when the use of an identity is verified, then local policy decisions about the mail can be made on the basis of the domain, rather than the host's IP address. This is advantageous because reputation of domain names is likely to be more accurate than reputation of host IP addresses. Furthermore, if a claimed identity fails verification, then local policy can take stronger action against such e-mail, such as rejecting it.

1.1 Protocol Status

SPF has been in development since the Summer of 2003, and has seen deployment beyond the developers beginning in December, 2003. The design of SPF has continuously evolved from them and is under active development today. There have been quite a number of forms of SPF, some written up as documents, some submitted as Internet Drafts, and many discussed and debated in development forums.

This document attempts to set down the common core of the SPF version 1 protocol, as implemented and deployed since about December, 2003. This conception of SPF is sometimes called "SPF Classic". The goal of this document is to be a stable reference that can be used for experimenting with existing implementations and developing SPF further. It is understood that particular implementations and deployments may differ from, and build upon, this work. It is hoped that we have nonetheless captured the common understanding of SPF version 1.

1.2 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This document is concerned with a portion of a mail message commonly called "envelope sender", "return path", "reverse path", "bounce address", "2821 from", or "mail from". Since these terms are either not well defined, or often used casually, this document defines the "Mail From" identity in Section 2.1. Note that other terms, that may superficially look like the common terms, such as "reverse-path" or "Return-Path" are used only with the defined meanings from normative documents.

2. Operation

2.1 The Mail From Identity

The "Mail From" identity derives from the SMTP MAIL command (see [RFC2821].) This command supplies the "reverse-path" for a message, which generally consists of the sender mailbox, and is the mailbox to which notification messages are sent if there are problems delivering the message.

This document defines the "Mail From" identity to be mailbox portion of the path of the reverse-path as defined in [RFC2821], Section 4.1.2. when it is non-null.

[RFC2821] allows the reverse-path to be null (see Section 4.5.5.) In this case, there is no explicit sender mailbox, and such a message can be assumed to be a notification message from the mail system itself. When the reverse-path is null, this document defines the "Mail From" identity to be the mailbox composed of the localpart "postmaster" and the domain supplied with the SMTP EHLO or HELO command. Note that requirements for the domain presented in the EHLO and HELO commands are not strict, and software must be prepared for a "Mail From" identity so constructed to be ill formed.

Generally, software that performs the authorization checks described below does so during a SMTP transaction, and so readily has the information required at hand.

2.2 Publishing Authorization

An SPF compliant domain name MUST publish a valid SPF record as described in Section 3. This record authorizes the use of the domain name in the envelope "Mail From" identity, by some sending MTAs, and not by others.

Domains SHOULD publish SPF records that end in "-all", or redirect to other records that do, so that a definitive determination of authorization can be made.

Domain holders may publish SPF records that explicitly authorize no hosts for domain names that shouldn't be used in sender mailboxes.

2.3 Checking Authorization

A mail receiver can perform an SPF compliant check for each mail message it receives. This check tests the authorization of a client host to inject mail with a given "Mail From" identity. Typically, such checks are done by a receiving MTA, but can be performed

elsewhere in the mail processing chain so long as the required information is available.

It is expected that mail receivers will use the SPF check as part of a larger set of tests on incoming mail. The results of other tests may influence whether or not a particular SPF check is performed. For example, finding the sending host on a local white list may cause all other tests to be skipped and all mail from that host to be accepted.

When a mail receiver decides to perform an SPF check, it MUST implement and evaluate the check_host() function (Section 4) correctly. While the test as a whole is optional, once it has been decided to perform a test it must be performed as specified so that the correct semantics are preserved between publisher and receiver.

To make the test, the mail receiver MUST evaluate the check_host() with the arguments set as follows:

<ip> - the IP address of the client host that is injecting the mail
<domain> - the domain portion of the "Mail From" identity
<sender> - the "Mail From" identity

Note that the <domain> argument may not be a well formed domain name. For example, if the reverse-path was null, then the EHLO or HELO domain is used, and that can be an address literal or entirely malformed in a valid SMTP transaction. In these cases, check_host() is defined in Section 4.3 to return a Fail result.

Software SHOULD perform this authorization check during the processing of the SMTP transaction that injects the mail. This allows errors to be returned directly to the injecting server by way of SMTP replies. Software can perform the check as early as the MAIL command, though it may be easier to delay the check to some later stage of the transaction.

Software can perform the authorization after the corresponding SMTP transaction has completed. There are two problems with this approach: 1) It may be difficult to accurately extract all the required information such as client IP address and HELO domain name. 2) If the authorization fails, then generating a non-delivery notification to the alleged sender is problematic as such an action would go against the explicit wishes of the alleged sender.

2.4 Interpreting the Result

The check_host() function returns one of seven results, some with

additional information. This section describes how software that performs the authorization must interpret the results. If the check is being performed during the SMTP mail transaction, it also describes how to respond.

2.4.1 Neutral

A Neutral result MUST be treated exactly like a None result.

2.4.2 Pass

A Pass result means that the client is authorized to inject mail with the given "Mail From" identity. Further policy checks, such as reputation, or black and/or white listing, can now proceed with confidence based on the "Mail From" identity.

2.4.3 Fail

A Fail result is an explicit statement that the client is not authorized to use the domain in the "Mail From" identity. The checking software can choose to mark the mail based on this, or to reject the mail outright.

If the checking software chooses to reject the mail during the SMTP transaction, then it MUST use a 550 reply code with an appropriate message. The Fail result includes a reason. The reason can be used to construct an appropriate message. If the reason is "Not Permitted", then an explanation string is also returned. This explanation string comes from the domain that published the SPF records and may contain a URL. Since that information doesn't originate with the checking software, the checking software will want to make it clear that text is not trusted. Example reply messages for rejecting are:

```
550 SPF Mail From check failed: Malformed Domain

550 SPF Mail From check failed: Domain Does Not Exist

550-SPF Mail From check failed: Not Permitted
550-The domain example.com said:
550 Please see http://www.example.com/mailpolicy.html
```

2.4.4 SoftFail

A SoftFail result should be treated as somewhere between a Fail and a Neutral. This value is used by domains as an intermediate state during roll-out of publishing records. The domain believes the host

isn't authorized but isn't willing to make that strong of a statement. Receiving software SHOULD NOT reject the message based on this result, but MAY subject the message to closer scrutiny.

2.4.5 None

A result of None means that no records were published by the domain. The checking software cannot ascertain if the client host is authorized or not.

2.4.6 TempError

A TempError result means that the receiving server encountered a transient error when performing the check. Checking software can choose to accept or temporarily reject the message. If the message is rejected during the SMTP transaction for this reason, the software MUST use a 450 reply code.

2.4.7 PermError

A PermError result means that the domain's published records couldn't be correctly interpreted for this "Mail From" identity. Checking software SHOULD reject the message. If rejecting during SMTP transaction time, a 550 reply MUST be used.

3. SPF Records

An SPF record declares which hosts are, and are not, authorized to use a domain name for the "Mail From" identity. Loosely, the record partitions all hosts into permitted and not-permitted sets. (Though some hosts might fall into other categories.)

The SPF record is a single string of text. An example record is:

```
v=spf1 +mx +a:colo.example.com/28 -all
```

This record has a version of "v=spf1" and three directives: "+mx", "+a:colo.example.com/28", and "-all".

3.1 Publishing

A domain name's SPF record is published in DNS. The record is placed in the DNS tree at the domain name it pertains to.

The previous example might be published easily via this line in a domain zone file:

```
example.com. IN SPF "v=spf1 +mx +a:colo.example.com/28 -all"
```

Note: The record is published at the domain name to which it pertains, not a name within the domain (such as is done with SRV records.) When published using the SPF RR type (see below), this poses no problems and was chosen as the clearest way to express the declaration. When published via TXT records it is still published directly at the domain name, even though other TXT records, for other purposes may be published there.

3.1.1 RR Types

This document defines a new DNS RR type SPF, type code to be determined. The format of this type is identical to the TXT RR [RFC1035].

However, because there are a number of DNS server and resolver implementations in common use that cannot handle new RR types, a record can be published with type TXT.

An SPF compliant domain name SHOULD have SPF records of both RR types. A compliant domain name MUST have a record of at least one type. If a domain has records of both types, they MUST have identical content.

An SPF compliant check SHOULD lookup both types. Lookups can be

performed serially or in parallel. If both types of records are obtained for a domain, the SPF type MUST be used.

It is recognized that the current practice (using a TXT type record), is not optimal, but a practical reality due to the state of deployed software. The two record type scheme provides a forward path to the better solution of using a RR type reserved for this purpose.

For either type, the character content of the record is encoded as US-ASCII.

Example RRs in this document are shown with the SPF record type, however they could also be published with a TXT type.

3.1.2 Multiple Records

A domain name MUST NOT have multiple records that would cause an authorization check to select more than one record. See Section 4.5 for the selection rules.

3.1.3 Additional Records

Some records contain directives that require additional SPF records. It is suggested that those records be placed under an "_spf" subdomain. See Appendix B for examples.

3.1.4 Multiple Strings

A Text DNS record (either TXT and SPF RR types) can be composed of more than one string. If a published record contains multiple strings, then the record MUST be treated as if those strings are concatenated together without adding spaces. For example:

```
SPF "v=spf1 ....  first" "second string..."
```

MUST be treated as equivalent to

```
SPF "v=spf1 ....  firstsecond string..."
```

SPF or TXT records containing multiple strings are useful in order to construct longer records which would otherwise exceed the maximum length of a string within a TXT or SPF RR record.

Note: Some nameserver implementations will silently split long strings in TXT records into several shorter strings.

3.1.5 Record Size

The published SPF record for a given domain name SHOULD remain small enough that the results of a query for it will fit within 512 octets. This will keep even older DNS implementations from falling over to TCP. Since the answer size is dependent on many things outside the scope of this document, it is only possible to give this guideline: If the combined length of the DNS name and the text of all the records of a given type (TXT or SPF) is under 480 characters, then DNS answers should fit in UDP packets. Note that when computing the sizes for queries of the TXT format, one must take into account any other TXT records published at the domain name.

3.1.6 Wildcard Records

Use of wildcard records for publishing is not recommended. Care must be taken if wildcard records are used. If a domain publishes wildcard MX records, it may want to publish wildcard declarations, subject to the same requirements and problems. In particular, the declaration must be repeated for any host that has any RR records at all, and for subdomains thereof. For example, the example given in [RFC1034], Section 4.3.3, could be extended with:

```
X.COM           MX      10      A.X.COM
X.COM           SPF     "v=spf1 +a:A.X.COM -all"

*.X.COM         MX      10      A.X.COM
*.X.COM         SPF     "v=spf1 +a:A.X.COM -all"

A.X.COM         A       1.2.3.4
A.X.COM         MX      10      A.X.COM
A.X.COM         SPF     "v=spf1 +a:A.X.COM -all"

*.A.X.COM       MX      10      A.X.COM
*.A.X.COM       SPF     "v=spf1 +a:A.X.COM -all"
```

Notice that SPF records must be repeated twice for every name within the domain: Once for the name, and once with a wildcard to cover the tree under the name.

Use of wildcards is discouraged in general as they cause every name under the domain to exist and queries against arbitrary names will never return RCODE 3 (Name Error).

4. The check_host() Function

The check_host() function fetches SPF records, parses them, and interprets them to evaluate if a particular host is or is not permitted to send mail with a given "Mail From" identity. Mail receivers that perform this check MUST correctly evaluate the check_host() function as described here.

Implementations MAY use a different algorithm than the canonical algorithm defined here, so long as the results are the same.

4.1 Arguments

The function check_host() takes three arguments:

```
<ip>     - the IP address of the host under test
<domain> - the domain to check
<sender> - the full sending mailbox address
```

The domain portion of <sender> will usually be the same as the <domain> argument when check_host() is initially evaluated. However, it will generally not be true for recursive evaluations (see Section 5.2 below).

Note: The IP address may be either IPv4 or IPv6.

4.2 Results

The function check_host() can result in one of seven results described here. Based on the result, the action to be taken is determined by and the local policies of the receiver. (see Section 2.4)

Results from interpreting valid records:

```
Neutral  (?): published data is explicitly inconclusive
Pass     (+): the <ip> is in the permitted set
Fail     (-): the <ip> is in the not permitted set
SoftFail (~): the <ip> may be in the not permitted set, its use is
              discouraged and the domain owner may move it to the not
              permitted set in the future
```

Results from error conditions:

```
None      - no published data
```

TempError - transient error during DNS lookup or other processing
PermError - unrecoverable error during processing, such as an error in the record format

If the result is "Fail", then an additional reason is returned. The reason may be one of:

Not Permitted
Malformed Domain
Domain Does Not Exist

If the reason is "Not Permitted", then an explanation string is also returned. The explanation string may be empty.

4.3 Initial Processing

If the <domain> is not an fully qualified domain name, check_host() immediately returns the result "Fail" and a reason of "Malformed Domain".

If the <sender> has no localpart, substitute the string "postmaster" for the localpart.

4.4 Record Lookup

The records for <domain> are fetched. If the records are in a cache, and have not expired, then they may simply be used. Otherwise, the records must be fetched from DNS as follows:

In accordance with how the records are published, (see Section 3.1 above), a DNS query needs to be made for the <domain> name, querying for either RR type TXT, SPF or both.

If the domain does not exist (RCODE 3), check_host() exits immediately with the result "Fail" and a reason of "Domain Does Not Exist"

If the DNS lookup returns a server failure (RCODE 2), or other error (RCODE other than 0 or 3), or the query times out, check_host() exits immediately with the result "TempError"

4.5 Selecting Records

Records begin with a version section:

```
record      = version terms *SP
version     = "v=spf1"
```

Starting with the set of records that were returned by the lookup, record selection proceeds in two steps:

1. If any records of type SPF are in the set, then all records of type TXT are discarded.
2. Records that do not begin with a version section of exactly "v=spf1" are discarded. Note that the version section is terminated either by a SP character or the end of the record. A record with a version section of "v=spf10" does not match and must be discarded.

After the above steps, there should be exactly one record remaining and evaluation can proceed. If there are no records remaining, check_host() exits immediately with the result "None". If there are two or more records remaining, then check_host() exits immediately with the error "PermError".

4.6 Record Evaluation

After one SPF record has been selected, the check_host() function parses and interprets it to find a result for the current test. If at any point a syntax error is encountered, check_host() returns immediately with the result "PermError".

Implementations MAY choose to parse the entire record first and return "PermError" if the record is not syntactically well formed. Note: Unrecognized mechanisms are still syntactically well formed. See Section 7.1.

4.6.1 Term Evaluation

There are two types of terms: mechanisms and modifiers. A given mechanism type may appear multiple times in a record. A given modifier may appear at most once per record. Unknown mechanisms cause processing to abort with the result "PermError". Unknown modifiers are ignored.

A record contains an ordered list of mechanisms and modifiers:

```
terms       = *( 1*SP ( directive / modifier ) )

directive   = [ prefix ] mechanism
prefix      = "+" / "-" / "?" / "~"
mechanism   = ( all / include
               / A / MX / PTR / IP4 / IP6 / exists
               / unknown-mechanism )
modifier    = name "=" macro-string
name        = alpha *( alpha / digit / "-" / "_" / "." )
```

Most mechanisms allow a ":" or "/" character after the name.

Modifiers always contain an equals ('=') character immediately after the name, and before any ":" or "/" characters that may be part of the macro-string.

Terms that do not contain any of "=", ":" or "/" are mechanisms.

Mechanism and modifier names are case-insensitive. A mechanism "INCLUDE" is equivalent to "include".

4.6.2 Mechanisms

Each mechanism is considered in turn from left to right.

When a mechanism is evaluated, one of three things can happen: it can match, it can not match, or it can throw an exception.

If it matches, processing ends and the prefix value is returned as the result of that record. (The default prefix value is "+".)

If it does not match, processing continues with the next mechanism. If no mechanisms remain, the default result is specified in Section 4.7.

If it throws an exception, mechanism processing ends and the exception value is returned.

The possible prefixes, and the results they return are:
"+" Pass
"-" Fail
"~" SoftFail
"?" Neutral

A missing prefix for a mechanism is the same as a prefix of "+".

When a mechanism matches, and the prefix is "-" so that a "Fail" result is returned, the reason is Not Permitted, and the explanation string is computed as described in Section 6.2.

Specific mechanisms are described in Section 5.

4.6.3 Modifiers

Modifiers are key/value pairs that affect the evaluation of the check_host() function.

The modifiers defined in this document ("redirect" and "exp") MAY

appear anywhere in the record, but SHOULD appear at the end, after all mechanisms. Ordering of these modifiers does not matter. These modifiers MUST NOT appear in a record more than once each. If they do, then check_host() exits with a result of "PermFail".

Unrecognized modifiers MUST be ignored no matter where in a record, or how often. This allows implementations of this document to handle records with modifiers that are defined in later versions.

4.7 Default result

If none of the mechanisms match and there is no "redirect" modifier, then the check_host() exits with a result of "Neutral". If there is a "redirect" modifier, check_host() proceeds as defined in Section 6.1.

Note that records SHOULD always either use a "redirect" modifier or an "all" mechanism to explicitly terminate processing.

For example:

```
   v=spf1 +mx -all
or
   v=spf1 +mx redirect=_spf.example.com
```

4.8 Domain Spec

Several of these mechanisms and modifiers have a <domain-spec> section. The <domain-spec> string is macro expanded (see Section 8). The resulting string is the common presentation form of a fully qualified DNS name: A series of labels separated by periods. This domain is called the <target-name> in the rest of this document.

Note: The result of the macro expansion is not subject to any further escaping. Hence, this facility cannot produce all characters that are legal in a DNS label, for example, the space or control characters. However, this facility is powerful enough to express legal host names, and common utility labels (such as "_spf") that are used in DNS.

For several mechanisms, the <domain-spec> is optional. If it is not provided, the <domain> is used as the <target-name>.

5. Mechanism Definitions

This section defines two types of mechanisms.

Basic mechanisms contribute to the language framework. They do not specify a particular type of authorization scheme.

- all
- include

Designated sender mechanisms are used to designate a set of <ip> addresses as being permitted or not to use the <domain> for sending mail.

- a
- mx
- ptr
- ip4
- ip6
- exists

Other mechanisms may be defined in the future.

The following conventions apply to all mechanisms that perform a comparison between <ip> and an IP address at any point:

If no CIDR-length is given in the directive, then <ip> and the IP address are compared for equality.

If a CIDR-length is specified, then only the specified number of high-order bits of <ip> and the IP address are compared for equality.

When any mechanism fetches host addresses to compare with <ip>, when <ip> is an IPv4 address, A records are fetched, when <ip> is an IPv6 address, AAAA records are fetched.

Several mechanisms rely on information fetched from DNS. For these DNS queries, if the DNS server returns an error (RCODE other than 0 or 3) or the query times out, the mechanism throws the exception "TempError". If the server returns "domain does not exist" (RCODE 3), then evaluation of the mechanism continues as if the server returned no error (RCODE 0) and zero answer records.

5.1 "all"

all = "all"

The "all" mechanism is a test that always matches. It is used as the

rightmost mechanism in a record to provide an explicit default.

For example:

```
v=spf1 +mx +a -all
```

Mechanisms after "all" will never be tested. Any "redirect" modifier (Section 6.1) has no effect when there is an "all" directive.

5.2 "include"

include = "include" ":" domain-spec

The "include" mechanism triggers a recursive evaluation of check_host(). The domain-spec is expanded as per Section 8. Then check_host() is evaluated with the resulting string as the <domain>. The <ip> and <sender> arguments remain the same as in the current evaluation of check_host().

"include" makes it possible for one domain to designate multiple administratively independent domains.

For example, a vanity domain "example.net" might send mail using the servers of administratively independent domains example.com and example.org.

Example.net could say

"v=spf1 include:example.com include:example.org -all".

That would direct check_host() to, in effect, check the records of example.com and example.org for a "pass" result. Only if the host were not permitted for either of those domains would the result be "Fail".

Whether this mechanism matches or not, or throws an error depends on the result of the recursive evaluation of check_host():

A recursive check_host() result of:	Causes the "include" mechanism to:
Pass	match
Fail	not match
SoftFail	not match
Neutral	not match

TempError	throw TempError
PermError	throw PermError
None	throw PermError

The "include" mechanism is intended for crossing administrative boundaries. While it is possible to use includes to consolidate multiple domains that share the same set of designated hosts, domains are encouraged to use redirects where possible, and to minimize the number of includes within a single administrative domain. For example, if example.com and example.org were managed by the same entity, and if the permitted set of hosts for both domains were "mx:example.com", it would be possible for example.org to specify "include:example.com", but it would be preferable to specify "redirect=example.com" or even "mx:example.com".

5.3 "a"

This mechanism matches if <ip> is one of the <target-name>'s IP addresses.

A = "a" [":" domain-spec] [dual-cidr-length]

An address lookup is done on the <target-name>. The <ip> is compared to the returned address(es). If any address matches, the mechanism matches.

5.4 "mx"

This mechanism matches if <ip> is one of the MX hosts for a domain name.

MX = "mx" [":" domain-spec] [dual-cidr-length]

check_host() first performs an MX lookup on the <target-name>. Then it performs an address lookup on each MX name returned. The <ip> is compared to each returned IP address. If any address matches, the mechanism matches.

Note Regarding Implicit MXes: If the <target-name> has no MX records, check_host() MUST NOT pretend the target is its single MX, and MUST NOT default to an A lookup on the <target-name> directly. This behavior breaks with the legacy "implicit MX" rule. See [RFC2821] Section 5. If such behavior is desired, the publisher should specify an "a" directive.

5.5 "ptr"

This mechanism tests if the DNS reverse mapping for <ip> exists and validly points to a domain name within a particular domain.

PTR = "ptr" [":" domain-spec]

First the <ip>'s name is looked up using this procedure: perform a DNS reverse-mapping for <ip>, looking up the corresponding PTR record in "in-addr.apra." if the address is an IPv4 one and "ip6.arpa." if it is an IPv6 address. For each record returned, validate the host name by looking up its IP address. If <ip> is among the returned IP addresses, then that host name is validated. In pseudocode:

```
sending-host_names := ptr_lookup(sending-host_IP);
for each name in (sending-host_names) {
  IP_addresses := a_lookup(name);
  if the sending-host_IP is one of the IP_addresses {
    validated_sending-host_names += name;
  }
}
```

Check all validated hostnames to see if they end in the <target-name> domain. If any do, this mechanism matches. If no validated hostname can be found, or if none of the validated hostnames end in the <target-name>, this mechanism fails to match.

Pseudocode:

```
for each name in (validated_sending-host_names) {
  if name ends in <domain-spec>, return match.
  if name is <domain-spec>, return match.
}
return no-match.
```

This mechanism matches if the <target-name> is an ancestor of a validated hostname, or if the <target-name> and a validated hostname are the same. For example: "mail.example.com" is within the domain "example.com", but "mail.bad-example.com" is not. If a validated hostname is the <target-name>, a match results.

Note: This mechanism is not recommended. If a domain decides to use it, it should make sure is has the proper PTR records in place for its hosts.

5.6 "ip4" and "ip6"

These mechanisms test if <ip> is contained within a given IP network.

```
IP4            = "ip4" ":" ip4-network [ ip4-cidr-length ]
IP6            = "ip6" ":" ip6-network [ ip6-cidr-length ]
ip4-cidr-length = "/" 1*DIGIT
ip6-cidr-length = "/" 1*DIGIT

ip4-network    = as per conventional dotted quad notation,
                 e.g. 192.0.2.0
ip6-network    = as per [RFC 3513], section 2.2,
                 e.g. 2001:DB8::CD30
```

The <ip> is compared to the given network. If CIDR-length high-order bits match, the mechanism matches.

If ip4-cidr-length is omitted it is taken to be "/32". If ip6-cidr-length is omitted it is taken to be "/128".

5.7 "exists"

This mechanism is used to construct an arbitrary host name that is used for a DNS A record query. It allows for complicated schemes involving arbitrary parts of the mail envelope to determine what is legal.

```
exists = "exists" ":" domain-spec
```

The domain-spec is expanded as per Section 8. The resulting domain name is used for a DNS A lookup. If any A record is returned, this mechanism matches. The lookup type is 'A' even when the connection type is IPv6.

Domains can use this mechanism to specify arbitrarily complex queries. For example, suppose example.com publishes the record:

```
v=spf1 exists:%{ir}.%{l1r+-}._spf.%{d} -all
```

The target-name might expand to "1.2.0.192.someuser._spf.example.com". This makes fine-grained decisions possible at the level of the user and client IP address.

This mechanism enables queries that mimic the style of tests that existing RBL lists use.

6. Modifier Definitions

Modifiers are not mechanisms: they do not return match or no-match. Instead they provide additional information or alter check_host() processing.

While unrecognized mechanisms cause an immediate "PermError" abort, unrecognized modifiers MUST be simply ignored. Modifiers therefore provide a way to extend the record format in the future with backward compatibility.

Only two modifiers are currently defined: "redirect" and "exp". Implementations of check_host() MUST support them both.

There is one deprecated modifier: "default", which cannot be defined by any future version of this document. Implementations MUST ignore it.

6.1 redirect: Redirected Query

If all mechanisms fail to match, and a "redirect" modifier is present, then processing proceeds as follows.

redirect = "redirect" "=" domain-spec

The domain-spec portion of the redirect section is expanded as per the macro rules in Section 8. Then check_host() is evaluated with the resulting string as the <domain>. The <ip> and <sender> arguments remain the same as current evaluation of check_host().

The result of this new evaluation of check_host() is then considered the result of the current evaluation.

Note that the newly queried domain may itself specify redirect processing.

This facility is intended for use by organizations that wish to apply the same record to multiple domains. For example:

```
  la.example.com. SPF "v=spf1 redirect=_spf.example.com"
  ny.example.com. SPF "v=spf1 redirect=_spf.example.com"
  sf.example.com. SPF "v=spf1 redirect=_spf.example.com"
_spf.example.com. SPF "v=spf1 mx:example.com -all"
```

In this example, mail from any of the three domains is described by the same record. This can be an administrative advantage.

Note: In general, a domain A cannot reliably use a redirect to

another domain B not under the same administrative control. Since the <sender> stays the same, there is no guarantee that the record at domain B will correctly work for addresses in domain A, especially if domain B uses mechanisms involving localparts. An "include" directive may be more appropriate.

For clarity it is RECOMMENDED that any "redirect" modifier appear as the very last term in a record.

6.2 exp: Explanation

explanation = "exp" "=" domain-spec

If check_host() results in a "Fail" due to a mechanism match (such as "-all"), and the "exp" modifier is present, then the explanation string returned is computed as described below. If no "exp" modifier is present, then an empty explanation string is returned.

The <domain-spec> is macro expanded (see Section 8) and becomes the <target-name>. The DNS TXT record for the <target-name> is fetched.

If <domain-spec> is empty, or there are any processing errors (any RCODE other than 0), or if no records are returned, or if more than one record is returned, then an empty explanation string is returned.

The fetched TXT record's strings are concatenated with no spaces, and then treated as a new macro-string which is macro-expanded. This final result is the explanation string.

Software evaluating check_host() can use this string when the result is "Fail" with a reason of "Not Permitted", to communicate information from the publishing domain in the form of a short message or URL. Software should make it clear that the explanation string comes from a third party. For example, it can prepend the macro string "%{d} explains: " to the explanation.

Implementations MAY limit the length of the resulting explanation string to allow for other protocol constraints and/or reasonable processing limits.

Suppose example.com has this record

```
v=spf1 mx -all exp=explain._spf.%{d}
```

Here are some examples of possible explanation TXT records at explain._spf.example.com:

```
      Example.com mail should only be sent by its own servers.

         -- a simple, constant message

      %{i} is not one of %{d}'s designated mail servers.

         -- a message with a little more info, including the IP address
            that failed the check

      See http://%{d}/why.html?s=%{S}&i=%{I}

         -- a complicated example that constructs a URL with the
            arguments to check_host() so that a web page can be
            generated with detailed, custom instructions
```

Note: During recursion into an "include" mechanism, explanations do not propagate out. But during execution of a "redirect" modifier, the explanation string from the target of the redirect is used.

7. Miscellaneous

7.1 Unrecognized Mechanisms and Modifiers

New mechanisms can only be introduced by new versions of this document.

Unrecognized mechanisms cause processing to abort: If, during evaluation of a record, check_host() encounters a mechanism which it does not understand, it terminates processing and returns "PermError", without evaluating any further mechanisms. Mechanisms listed before the unknown mechanism MUST, however, be evaluated.

For example, consider the record:

```
v=spf1 a mx ptr foo:_foo.%{d} -all
```

If during the evaluation of check_host(), any of the "a", "mx", or "ptr" directives match, then check_host() would return a "Pass" result. If none of those directives resulted in a match, then an implementation that did not recognize the "foo" mechanism would return "PermError". An implementation that did recognize the "foo" mechanism would be able to perform an extended evaluation.

Note: "foo" is an example of an unknown extension mechanism that could be defined in the future. It is NOT defined by this proposal.

Unrecognized modifiers are ignored: if an implementation encounters modifiers which it does not recognize, it MUST ignore them.

7.2 Processing Limits

During processing, an evaluation of check_host() may require additional evaluations of check_host() due to the "include" mechanism and/or the "redirect" modifier.

Implementations must be prepared to handle records that are set up incorrectly or maliciously. Implementations MUST perform loop detection, limit additional evaluations, or both. If an implementation chooses to limit additional evaluations, then at least a total of 10 evaluations of check_host() for a single query MUST be supported. (This number should be enough for even the most complicated configurations.)

If a loop is detected, or the evaluation limit of an implementation is reached, check_host() MUST abort processing and return the result "PermError".

MTAs or other processors MAY also impose a limit on the maximum amount of elapsed time to evaluate check_host(). Such a limit SHOULD allow at least 20 seconds. If such a limit is exceeded, the result of authentication SHOULD be "TempError".

Domains publishing records SHOULD try to keep the number of "include" directives and chained "redirect" modifiers to a minimum. Domains SHOULD also try to minimize the amount of other DNS information needed to evaluate a record. This can be done by choosing directives that require less DNS information.

For example, consider a domain set up as:

```
example.com.      IN MX   10 mx.example.com.
mx.example.com.   IN A    192.0.2.1
a.example.com.    IN SPF  "v=spf1 +mx:example.com -all"
b.example.com.    IN SPF  "v=spf1 +a:mx.example.com -all"
c.example.com.    IN SPF  "v=spf1 +ip4:192.0.2.1 -all"
```

Evaluating check_host() for the domain "a.example.com" requires the MX records for "example.com", and then the A records for the listed hosts. Evaluating for "b.example.com" only requires the A records. Evaluating for "c.example.com" requires none.

However, there may be administrative considerations: Using "a" over "ip4" allows hosts to be renumbered easily. Using "mx" over "a" allows the set of mail hosts to be changed easily.

8. Macros

8.1 Macro definitions

Many mechanisms and modifiers perform macro interpolation on part of the term.

```
domain-spec   = *( macro-expand / macro-literal )
macro-string  = *( macro-expand / macro-literal / "/" )
macro-expand  = ( "%{" ALPHA transformer *delimiter "}" )
                / "%%" / "%_" / "%-"
macro-literal = %x21-24 / %x26-2E / %x30-7E
              ; visible characters except "%" and "/"
transformer   = *DIGIT [ "r" ]
delimiter     = "." / "-" / "+" / "," / "/" / "_" / "="
```

A literal "%" is expressed by "%%".
"%_" expands to a single " " space.
"%-" expands to a URL-encoded space, viz. "%20".

The following macro letters are expanded in term arguments:

- s = <sender>
- l = local-part of <sender>
- o = domain of <sender>
- d = <domain>
- i = <ip>
- p = the validated host name of <ip>
- v = the string "in-addr" if <ip> is ipv4, or "ip6" if <ip> is ipv6

The following macro letters are only allowed in "exp" text:

- c = SMTP client IP (easily readable format)
- r = domain name of host performing the check
- t = current timestamp

The uppercase versions of all these macros are URL-encoded.

A '%' character not followed by a '{', '%', '-', or '_' character MUST be interpreted as a literal. Domains SHOULD NOT rely on this feature; they MUST escape % literals. For example, an explanation TXT record

```
   Your spam volume has increased by 581%
```

is incorrect. Instead, say

Your spam volume has increased by 581%%

All other legal visible characters are simply expanded to themselves. Note that the two different macro contexts, domain-spec, and macro-string allow slightly different sets of legal visible characters. In particular, macro-string allows the slash character.

Legal optional transformers are:

*DIGIT : zero or more digits
'r' : reverse value, splitting on dots by default

If transformers or delimiters are provided, the replacement value for a macro letter is split into parts. After performing any reversal operation and/or removal of left-hand parts, the parts are rejoined using "." and not the original splitting characters.

By default, strings are split on "." (dots). Note that no special treatment is given to leading, trailing or consecutive delimiters, and so the list of parts may contain empty strings. Macros may specify delimiter characters which are used instead of ".". Delimiters MUST be one or more of the characters:

"." / "-" / "+" / "," / "/" / "_" / "="

The 'r' transformer indicates a reversal operation: if the client IP address were 192.0.2.1, the macro %{i} would expand to "192.0.2.1" and the macro %{ir} would expand to "1.2.0.192".

The DIGIT transformer indicates the number of right-hand parts to use, after optional reversal. If a DIGIT is specified, the value MUST be nonzero. If no DIGITs are specified, or if the value specifies more parts than are available, all the available parts are used. If the DIGIT was 5, and only 3 parts were available, the macro interpreter would pretend the DIGIT was 3. Implementations MUST support at least a value of 128, as that is the maximum number of labels in a domain name.

The "s" macro expands to the <sender> argument. It is an e-mail address with a localpart, an "@" character, and a domain. The "l" macro expands to just the localpart. The "o" macro expands to just the domain part. Note that these values remain the same during recursive and chained evaluations due to "include" and/or "redirect". Note also that if the original <sender> had no localpart, the localpart was set to "postmaster" in initial processing (see Section 4.3).

For IPv4 addresses, both the "i" and "c" macros expand to the standard dotted-quad format.

For IPv6 addresses, the "i" macro expands to a dot-format address; it is intended for use in %{ir}. The "c" macro may expand to any of the hexadecimal colon-format addresses specified in [RFC3513] section 2.2. It is intended for humans to read.

The "p" macro expands to the validated host name of <ip>. The procedure for finding the validated host names is defined in Section 5.5. If that procedure produces more than one validated host name, any name from the list may be used. If that procedure produces no validate host name the string "unknown" is used.

The "r" macro expands to the name of the receiving MTA. This SHOULD be a fully qualified domain name, but if one does not exist (as when the checking is done by a script) or if policy restrictions dictate otherwise, the word "unknown" SHOULD be substituted. The domain name may be different than the name found in the MX record that the client MTA used to locate the receiving MTA.

The "t" macro expands to the decimal representation of the number of seconds since the Epoch (Midnight, January 1st, 1970, UTC). This is the same value as returned by the time() function in most standards compliant libraries.

Any unrecognized macro letters are expanded as the string "unknown". There is one deprecated macro letter: "h". It is expanded as the string "deprecated".

When the result of macro expansion is used in a domain name query, if the expanded domain name exceeds 255 characters (the maximum length of a domain name), the left side is truncated to fit, by removing successive subdomains until the total length falls below 255 characters.

Uppercased macros expand exactly as their lower case equivalents, and are then URL escaped. URL escaping is described in [RFC2396].

Note: Domains should avoid using the "s", "l" or "o" macros in conjunction with any mechanism directive. While these macros are powerful and allow per-user records to be published, they severely limit the ability of implementations to cache results of check_host() and they reduce the effectiveness of DNS caches.

Implementations should be aware that if no directive processed during the evaluation of check_host() contains an "s", "l", or "o" macro, then the results of the evaluation can be cached on the basis of

<domain> and <ip> alone for as long as the shortest TTL of all the DNS records involved.

8.2 Expansion Examples

The <sender> is strong-bad@email.example.com.
The IPv4 SMTP client IP is 192.0.2.3.
The IPv6 SMTP client IP is 5f05:2000:80ad:5800::1.
The PTR domain name of the client IP is mx.example.org.

macro	expansion
%{s}	strong-bad@email.example.com
%{o}	email.example.com
%{d}	email.example.com
%{d4}	email.example.com
%{d3}	email.example.com
%{d2}	example.com
%{d1}	com
%{dr}	com.example.email
%{d2r}	example.email
%{l}	strong-bad
%{l-}	strong.bad
%{lr}	strong-bad
%{lr-}	bad.strong
%{l1r-}	strong

macro-string	expansion
%{ir}.%{v}._spf.%{d2}	3.2.0.192.in-addr._spf.example.com
%{lr-}.lp._spf.%{d2}	bad.strong.lp._spf.example.com
%{lr-}.lp.%{ir}.%{v}._spf.%{d2}	bad.strong.lp.3.2.0.192.in-addr._spf.example.com
%{ir}.%{v}.%{l1r-}.lp._spf.%{d2}	3.2.0.192.in-addr.strong.lp._spf.example.com
%{d2}.trusted-domains.example.net	example.com.trusted-domains.example.net
IPv6:	
%{ir}.%{v}._spf.%{d2}	1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.5.d.a.0.8.0.0.0.2.5.0.f.5.ip6._spf.example.com

9. Implications

This section outlines the major implications that adoption of this document will have on various entities involved in Internet e-mail. It is intended to make clear to the reader where this document knowingly affects the operation of such entities. This section is not a "how-to" manual, nor a "best practices" document, and is not a comprehensive list of what such entities should do in light of this document.

This section is non-normative.

9.1 Sending Domains

Domains that wish to be compliant with this specification will need to determine the list of hosts that they allow to use their domain name in the "Mail From" identity. It is recognized that forming such a list is not just a simple technical exercise, but involves policy decisions with both technical and administrative considerations.

9.2 Mailing Lists

Mailing lists must be aware of how they re-inject mail that is sent to the list. If the list re-injects mail with the same reverse-path that the mail had when it was received, then that mail may fail the authorization tests defined in this document. In particular, they will fail when the domain of the reverse-path publishes SPF records for the "Mail From" identity, those records do not authorize the mailing list host, and a receiver of the mailing list performs the authorization test.

Almost all mailing list software in use for public mailing lists uses a reverse-path with the mailing list's own domain so that the software can receive mail bounces and assist in the administration of the list. Lists that use such software, configured to operate this way will require only one modest change in light of this document: The mailing list host needs to be authorized by the mailing list domain's own SPF record, if the domain publishes one.

Mailing lists based on simple alias expansion, or other software that doesn't manage bounces directly, may or may not encounter problems depending on how access to the list is restricted. Such lists that are entirely internal to a domain (only people in the domain can send to or receive from the list) are not affected.

9.3 Forwarding Services

Forwarding services take mail that is received at a mailbox and

direct it to some external mailbox. At the time of this writing, the near-universal practice of such services is to use the original reverse-path of a message when re-injecting it for delivery to the external mailbox. This means the external mailbox's MTA sees all such mail in a connection from a host of the forwarding service, and so the "Mail From" identity will not in general pass authorization.

There are several possible ways that this authorization failure can be ameliorated. If the owner of the external mailbox wishes to trust the forwarding service, they can direct the external mailbox's MTA to skip such tests when the client host belongs to the forwarding service. Tests against some other identity may also be used to override the test against the "Mail From" identity.

For larger domains, it may not be possible to have a complete or accurate list of forwarding services used by the owners of the domain's mailboxes. In such cases, white lists of generally recognized forwarding services could be employed.

Forwarding services could also skirt the issue by using reverse-paths that contain their own domain. This means that mail bounced from the external mailbox will have to be re-bounced by the forwarding service. Various schemes to do this exist though they vary widely in complexity and resource requirements on the part of the forwarding service.

9.4 Mail Services

Entities that offer mail services to other domains such as sending of bulk mail will may have to alter their mail in light of the authorization check in this document. If the reverse-path used for such e-mail uses the domain of the mail service provider, then the provider needs only to ensure that their sending host is authorized by their own SPF record, if any.

If the reverse-path does not use the mail service provider's domain, then extra care must be taken. The SPF record format has several options for authorizing the sending MTAs of another domain (the service provider's)

9.5 MTA Relays

The authorization check generally precludes the use of arbitrary MTA relays between sender and receiver of an e-mail message.

Within an organization, MTA relays can be effectively deployed. However, for purposes of this document, such relays are effectively invisible. The "Mail From" identity authorization check is a check

between border MTAs.

For mail senders, this means that published SPF records must authorize any MTAs that actually send across the Internet. Usually, these are just the border MTAs as internal MTAs simply forward mail to these MTAs for delivery.

Mail receivers will generally want to perform the authorization check at the border MTAs. This allows mail that fails to be rejected during the SMTP session rather than bounced. Internal MTAs then do not perform the authorization test. To perform the authorization test other than at the border, the host that first transferred the message to the organization must be determined, which can be difficult to extract from headers. Testing other than at the border is not recommended.

10. Security Considerations

The "Mail From" identity authorization must not be construed to provide more assurance than it does. It is entirely possible for a malicious sender to inject a message with a reverse-path that uses their own domain, to have that domain's SPF record authorize the sending host, and yet the message content can easily claim other identities in the headers. Unless the user, or the MUA takes care to note that the authorized "Mail From" identity does not match the other, more commonly presented identities (such as the From: header), the user may be lulled into a false sense of security.

There are two aspects of this protocol that malicious parties could exploit to undermine the validity of the check_host() function:

- The evaluation of check_host() relies heavily on DNS. A malicious attacker could attack the DNS infrastructure and cause check_host() to see spoofed DNS data, and then return incorrect results. This could include returning "Pass" for an <ip> value where the actual domain's record would evaluate to "Fail". See [RFC3833] for a description of the DNS weaknesses.
- The client IP address, <ip>, is assumed to be correct. A malicious attacker could spoof TCP sequences to make mail appear to come from a permitted host for a domain that the attacker is impersonating.

As with most aspects of mail, there are a number of ways that malicious parties could use the protocol as an avenue of a distributed denial of service attack:

- While implementations of check_host() need to limit the number of "include" and "redirect" terms and/or check for loops, malicious domains could publish records that exercise or exceed these limits in an attempt to waste computation effort at their targets when they send them mail.
- Malicious parties could send large volume mail purporting to come from the intended target to a wide variety of legitimate mail hosts. These legitimate machines would then present a DNS load on the target as they fetched the relevant records.
- While these distributed denial of service attacks are possible, they seem more convoluted to mount, and have less of an impact, than other simpler attacks.

When the authorization check fails with the code "Not Permitted", an explanation string may be included in the reject response. Both the sender and the rejecting receiver need to be aware that the explanation was determined by the publisher of the SPF record checked, and is in general not the receiver. The explanation may

contain URLs that may be malicious, and/or offensive or misleading text. This is probably less of a concern than it may seem since such messages are returned to the sender, and their source is the SPF record published by the domain in the "Mail From" identity claimed by that very sender. To put it another way, the only people who see malicious explanation strings are people whose messages claim to be from domains that publish such strings in their SPF records.

11. IANA Considerations

The IANA needs to assign a new Resource Record Type and Qtype from the DNS Parameters Registry for the SPF RR type.

12. Contributors and Acknowledgements

This design owes a debt of parentage to [RMX] by Hadmut Danisch and to [DMP] by Gordon Fecyk. The idea of using a DNS record to check the legitimacy of an email address traces its ancestry farther back through messages on the namedroppers mailing list by Paul Vixie [Vixie] (based on suggestion by Jim Miller) and by David Green [Green].

Philip Gladstone contributed macros to the specification, multiplying the expressiveness of the language and making per-user and per-IP lookups possible.

The authors would also like to thank the literally hundreds of individuals who have participated in the development of this design. There are far too numerous to name, but they include:

The folks on the SPAM-L mailing list.
The folks on the ASRG mailing list.
The folks on the spf-discuss mailing list.
The folks on #perl.
The folks in the MARID working group and on the MXCOMP mailing list.

13. References

13.1 Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[RFC2396] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[RFC2821] Klensin, J., "Simple Mail Transfer Protocol", RFC 2821, April 2001.

[RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.

13.2 Informative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC3833] Atkins, D. and R. Austein, "Threat Analysis of the Domain Name System (DNS)", RFC 3833, August 2004.

[RMX] Danish, H., "The RMX DNS RR Type for light weight sender authentication", October 2003.

Work In Progress

[DMP] Fecyk, G., "Designated Mailers Protocol", December 2003.

Work In Progress

[Vixie] Vixie, P., "Repudiating Mail-From", 2002.

[Green] Green, D., "Domain-Authorized SMTP Mail", 2002.

Authors' Addresses

Mark Lentczner
1209 Villa Street
Mountain View, CA 94041
United States of America

EMail: markl@glyphic.com
URI: http://www.ozonehouse.com/mark/

Meng Weng Wong
Singapore

EMail: mengwong+spf@pobox.com
URI: http://spf.pobox.com/

Appendix A. Collected ABNF

This section is normative and any discrepancies with the ABNF fragments in the preceding text are to be resolved in favor of this grammar.

See [RFC2234] for ABNF notation. Please note that as per this ABNF definition, literal text strings (those in quotes) are case insensitive. Hence, "mx" matches "mx", "MX", "mX" and "Mx".

```
record      = version terms *SP
version     = "v=spf1"

terms       = *( 1*SP ( directive / modifier ) )

directive   = [ prefix ] mechanism
prefix      = "+" / "-" / "?" / "~"
mechanism   = ( all / include
              / A / MX / PTR / IP4 / IP6 / exists
              / unknown-mechanism )

all          = "all"
include      = "include"  ":" domain-spec
A            = "a"      [ ":" domain-spec ] [ dual-cidr-length ]
MX           = "mx"     [ ":" domain-spec ] [ dual-cidr-length ]
PTR          = "ptr"    [ ":" domain-spec ]
IP4          = "ip4"      ":" ip4-network   [ ip4-cidr-length ]
IP6          = "ip6"      ":" ip6-network   [ ip6-cidr-length ]
exists       = "exists"   ":" domain-spec

unknown-mechanism = name [ ":" macro-string ]

modifier    = redirect / explanation / unknown-modifier
redirect    = "redirect" "=" domain-spec
explanation = "exp" "=" domain-spec
unknown-modifier = name "=" macro-string


ip4-network  = as per conventional dotted quad notation,
               e.g. 192.0.2.0
ip6-network  = as per [RFC 3513], section 2.2,
               e.g. 2001:DB8::CD30

dual-cidr-length = [ ip4-cidr-length ] [ "/" ip6-cidr-length ]
ip4-cidr-length  = "/" 1*DIGIT
ip6-cidr-length  = "/" 1*DIGIT

domain-spec   = *( macro-expand / macro-literal )
```

```
   macro-string  = *( macro-expand / macro-literal / "/" )
   macro-expand  = ( "%{" ALPHA transformer *delimiter "}" )
                   / "%%" / "%_" / "%-"
   macro-literal = %x21-24 / %x26-2E / %x30-7E
                 ; visible characters except "%" and "/"
   transformer   = *DIGIT [ "r" ]
   delimiter     = "." / "-" / "+" / "," / "/" / "_" / "="

   name = ALPHA *( ALPHA / DIGIT / "-" / "_" / "." )
```

Appendix B. Extended Examples

These examples are based on the following DNS setup:

```
; A domain with two mail servers, two hosts
; and two servers at the domain name

$ORIGIN example.com.
@           MX  10 mail-a
            MX  20 mail-b
            A   192.0.2.10
            A   192.0.2.11
amy         A   192.0.2.65
bob         A   192.0.2.66
mail-a      A   192.0.2.129
mail-b      A   192.0.2.130
www         CNAME example.com.

; A related domain

$ORIGIN example.org
@           MX  10 mail-c
mail-c      A   192.0.2.140

; The reverse IP for those addresses

$ORIGIN 2.0.192.in-addr.arpa.
10          PTR example.com.
11          PTR example.com.
65          PTR amy.example.com.
66          PTR bob.example.com.
129         PTR mail-a.example.com.
130         PTR mail-b.example.com.
140         PTR mail-c.example.org.

; A rogue reverse IP domain that claims to be
; something it's not

$ORIGIN 0.0.10.in-addr.arpa.
4           PTR bob.example.com.
```

B.1 Simple Examples

These examples show various possible published records for example.com and which values if <ip> would cause check_host() to return "Pass". Note that <domain> is "example.com".

```
   v=spf1 +all
      -- any <ip> passes

   v=spf1 a -all
      -- hosts 192.0.2.10 and 192.0.2.11 pass

   v=spf1 a:example.org -all
      -- no sending hosts pass since example.org has no A records

   v=spf1 mx -all
      -- sending hosts 192.0.2.129 and 192.0.2.130 pass

   v=spf1 mx:example.org -all
      -- sending host 192.0.2.140 passes

   v=spf1 mx mx:example.org -all
      -- sending hosts 192.0.2.129, 192.0.2.130, and 192.0.2.140 pass

   v=spf1 mx/30 mx:example.org/30 -all
      -- any sending host in 192.0.2.128/30 or 192.0.2.140/30 passes

   v=spf1 ptr -all
      -- sending host 192.0.2.65 passes (reverse IP is valid and in
      example.com)
      -- sending host 192.0.2.140 fails (reverse IP is valid, but not in
      example.com)
      -- sending host 10.0.0.4 fails (reverse IP is not valid)

   v=spf1 ip4:192.0.2.128/28 -all
      -- sending host 192.0.2.65 fails
      -- sending host 192.0.2.129 passes
```

B.2 Multiple Domain Example

These examples show the effect of related records:

```
   example.org: "v=spf1 include:example.com include:example.net -all"
```

This record would be used if mail from example.org actually came through servers at example.com and example.net. Example.org's designated servers are the union of example.com and example.net's designated servers.

```
   la.example.org: "v=spf1 redirect=example.org"
   ny.example.org: "v=spf1 redirect=example.org"
   sf.example.org: "v=spf1 redirect=example.org"
```

These records allow a set of domains that all use the same mail

system to make use of that mail system's record. In this way, only the mail system's record needs to updated when the mail setup changes. These domains' records never have to change.

B.3 RBL Style Example

Imagine that, in addition to the domain records listed above, there are these:

```
$Origin _spf.example.com.
mary.mobile-users                   A 127.0.0.2
fred.mobile-users                   A 127.0.0.2
15.15.168.192.joel.remote-users     A 127.0.0.2
16.15.168.192.joel.remote-users     A 127.0.0.2
```

The following records describe users at example.com who mail from arbitrary servers, or who mail from personal servers.

example.com:

```
v=spf1 mx
       include:mobile-users._spf.%{d}
       include:remote-users._spf.%{d}
       -all
```

mobile-users._spf.example.com:

```
v=spf1 exists:%{l1r+}.%{d}
```

remote-users._spf.example.com:

```
v=spf1 exists:%{ir}.%{l1r+}.%{d}
```

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.