**Sender Policy Framework (SPF) for Authorizing Use of Domains in Email,
Version 1
draft-ietf-spfbis-4408bis-11.txt**

Abstract

   Email on the Internet can be forged in a number of ways. In
   particular, existing protocols place no restriction on what a sending
   host can use as the "MAIL FROM" of a message or the domain given on
   the SMTP HELO/EHLO commands. This document describes version 1 of
   the Sender Policy Framework (SPF) protocol, whereby an ADMD can
   explicitly authorize the hosts that are allowed to use its domain
   names, and a receiving host can check such authorization.

   This document obsoletes RFC4408.

Status of this Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on August 29, 2013.

Copyright Notice

Table of Contents

1. Introduction

The current email infrastructure has the property that any host
injecting mail into the system can use any DNS domain name it wants
in each of the various identifiers specified by [RFC5321] and
[RFC5322]. Although this feature is desirable in some circumstances,
it is a major obstacle to reducing Unsolicited Bulk Email (UBE, aka
spam). Furthermore, many domain owning ADMDs (ADministrative
Management Domains, see [RFC5598]) are understandably concerned about
the ease with which other entities can make use of their domain
names, often with malicious intent.

This document defines a protocol by which ADMDs can authorize hosts
to use their domain names in the "MAIL FROM" or "HELO" identities.
Compliant ADMDs publish Sender Policy Framework (SPF) records in the
DNS specifying which hosts are permitted to use their names, and
compliant mail receivers use the published SPF records to test the
authorization of sending Mail Transfer Agents (MTAs) using a given
"HELO" or "MAIL FROM" identity during a mail transaction.

An additional benefit to mail receivers is that after the use of an
identity is verified, local policy decisions about the mail can be
made based on the sender's 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, local policy can take
stronger action against such email, such as rejecting it.

1.1. Terminology

1.1.1. Keywords

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

1.1.2. Imported Definitions

The ABNF tokens "ALPHA", "DIGIT", and "SP" are defined in [RFC5234].

The token "local-part" is defined in [RFC5321].

"dot-atom", "quoted-string", "comment", "CFWS", "FWS", and "CRLF" are
defined in [RFC5322].

1.1.3. MAIL FROM Definition

This document is concerned with the portion of a mail message
commonly called "envelope sender", "return path", "reverse path",
"bounce address", "5321 FROM", "MAIL FROM", or RFC5321.MailFrom.
Since these terms are either not well defined or often used casually,
this document uses "MAIL FROM" for consistency. This means the
RFC5321.MailFrom as defined in [RFC5598]. Note that other terms that
might superficially look like the common terms, such as "reverse-
path", are used only with the defined meanings from normative
documents.

1.1.4. HELO Definition

This document also makes use of the HELO/EHLO identity. The "HELO"
identity derives from either the SMTP HELO or EHLO command (see
[RFC5321]). Since HELO and EHLO can, in many cases, be used
interchangeably, they are identified commonly as "HELO" in this
document. This means RFC5321.HELO/.EHLO as defined in [RFC5598].
These commands supply the identity of the SMTP client (sending host)
for the SMTP session.

2. Operational Overview

2.1. The "HELO" Identity

   It is RECOMMENDED that SPF verifiers not only check the "MAIL FROM"
   identity, but also separately check the "HELO" identity by applying
   the check_host() function (Section 4) to the "HELO" identity as the
   <sender>. Checking "HELO" promotes consistency of results and can
   reduce DNS resource usage. Additionally, since SPF records published
   for "HELO" identities refer to a single host, when available, they
   are a very reliable source of host authorization status.

   Note that requirements for the domain presented in the EHLO or HELO
   command are not always clear to the sending party, and SPF verifiers
   MUST be prepared for the "HELO" identity to be malformed or an IP
   address literal. This SPF check can only be performed when the
   "HELO" string is a valid fully qualified domain.

2.2. The "MAIL FROM" Identity

   SPF verifiers MUST check the ""MAIL FROM" identity if a completed
   "HELO" check has not reached a definitive policy result by applying
   the check_host() function to the "MAIL FROM" identity as the
   <sender>.

   [RFC5321] allows the reverse-path to be null (see Section 4.5.5 in
   [RFC5321]). 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
   local-part "postmaster" and the "HELO" identity (which might or might
   not have been checked separately before).

2.3. Publishing Authorization

   An SPF-compliant domain MUST have valid SPF records as described in
   Section 3. These records authorize the use of the relevant domain
   names in the "HELO" and "MAIL FROM" identities by the MTAs specified
   therein.

   SPF results can be used to make both positive (source is authorized)
   and negative (source is not authorized) determinations. If domain
   owners choose to publish SPF records and want to support receivers
   making negative authorization determinations, then they MUST publish
   records that end in "-all", or redirect to other records that do,
   otherwise, no definitive determination of authorization can be made.
   Potential issues and mitigations associated with negative
   determinations are discussed in Section 10.

ADMDs can publish SPF records that explicitly authorize no hosts for
domain names that are neither used in the domain part of email
addresses nor expected to originate mail.

When changing SPF records, care has to be taken to ensure that there
is a transition period so that the old policy remains valid until all
legitimate email can reasonably expect to have been checked.
[RFC5321] Section 4.5.4.1 discusses how long a message might be in
transit. While offline checks are possible, the closer to the
original transmission time checks are performed, the more likely they
are to get an SPF result that matches the sending ADMD intent at the
time the message was sent.

2.4. Checking Authorization

A mail receiver can perform a set of SPF checks for each mail message
it receives. An SPF check tests the authorization of a client host
to emit mail with a given 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 and
reliable. At least the "MAIL FROM" identity MUST be checked, but it
is RECOMMENDED that the "HELO" identity also be checked beforehand.

Without explicit approval of the domain owner, checking other
identities against SPF version 1 records is NOT RECOMMENDED because
there are cases that are known to give incorrect results. For
example, almost all mailing lists rewrite the "MAIL FROM" identity
(see Section 10.3.1), but some do not change any other identities in
the message. The scenario described in Section 10.3.2, sub-section
1.2, is another example. Documents that define other identities will
have to define the method for explicit approval.

It is possible that mail receivers will use the SPF check as part of
a larger set of tests on incoming mail. The results of other tests
might influence whether or not a particular SPF check is performed.
For example, finding the sending host's IP address on a local white
list might 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 use a
correctly-implemented check_host() function (Section 4) evaluated
with the correct parameters. Although the test as a whole is
optional, once it has been decided to perform a test it has to 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()
function with the arguments set as follows:

   <ip>   - the IP address of the SMTP client that is emitting the
              mail, either IPv4 or IPv6.

   <domain> - the domain portion of the "MAIL FROM" or "HELO" identity.

   <sender> - the "MAIL FROM" or "HELO" identity.

   Although invalid, malformed, or non-existent domains cause SPF checks
   to return "none" because no SPF record can be found, it has long been
   the policy of many MTAs to reject email from such domains, especially
   in the case of invalid "MAIL FROM". Rejecting email will prevent one
   method of circumventing of SPF records.

   Implementations have to take care to correctly extract the <domain>
   from the data given with the SMTP MAIL FROM command as many MTAs will
   still accept such things as source routes (see [RFC5321], Appendix
   C), the %-hack (see [RFC1123]), and bang paths (see [RFC1983]).
   These archaic features have been maliciously used to bypass security
   systems.

2.5. Location of Checks

   The authorization check SHOULD be performed during the processing of
   the SMTP transaction that sends the mail. This reduces the
   complexity of determining the correct IP address to use as an input
   to check_host() and allows errors to be returned directly to the
   sending MTA by way of SMTP replies.

   Performing the authorization other than using the return-path and
   client address at the time of the MAIL command during the SMTP
   transaction can cause problems, such as the following: (1) It might
   be difficult to accurately extract the required information from
   potentially deceptive headers; (2) legitimate email might fail
   because the sender's policy had since changed.

   Generating non-delivery notifications to forged identities that have
   failed the authorization check is a source of backscatter and SHOULD
   be avoided. Section 2 of [RFC3834] describes backscatter and the
   problems it causes.

2.6. Results of Evaluation

   Section 4 defines check_host(), a model function definition that uses
   the inputs defined above and the sender's policy published in the DNS
   to reach a conclusion about client authorization. An SPF verifier
   implements something semantically identical to the function defined
   there.

This section enumerates and briefly defines the possible outputs of
that function. Information about how to handle these outputs is in
Section 8.

2.6.1. None

A result of "none" means either (a) no syntactically valid DNS domain
name was extracted from the SMTP session that could be used as the
one to be authorized, or (b) no TXT records were retrieved from the
DNS that appeared to be intended for use by SPF verifiers.

2.6.2. Neutral

The domain owner has explicitly stated that it is not asserting
whether the IP address is authorized. This result MUST be treated
exactly like the "none" result; the distinction exists only for
informational purposes.

2.6.3. Pass

A "pass" result means that the client is authorized to inject mail
with the given identity. The domain can now, in the sense of
reputation, be considered responsible for sending the message.
Further policy checks can now proceed with confidence in the
legitimate use of the identity. This is further discussed in
Appendix G.1.

2.6.4. Fail

A "fail" result is an explicit statement that the client is not
authorized to use the domain in the given identity.

2.6.5. Softfail

The domain owner has published a weak statement that the host is
probably not authorized. It has not published a stronger, more
definitive policy that results in a "fail"

2.6.6. Temperror

A "temperror" result means the SPF verifier encountered a transient
(generally DNS) error while performing the check.

2.6.7. Permerror

A "permerror" result means the domain's published records could not
be correctly interpreted. This signals an error condition that
definitely requires manual intervention to be resolved.

3. SPF Records

   An SPF record is a DNS record that declares which hosts are, and are
   not, authorized to use a domain name for the "HELO" and "MAIL FROM"
   identities. Loosely, the record partitions all hosts into permitted
   and not-permitted sets (though some hosts might fall into neither
   category).

   The SPF record is a single string of text. The record format is
   described below in Section 4. An example record is the following:

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

   This record has a version of "spf1" and three directives: "+mx",
   "a:colo.example.com/28" (the + is implied), and "-all".

   Each SPF record is placed in the DNS tree at the host name it
   pertains to, not a subdomain under it, such as is done with SRV
   records [RFC2782].

   The example in this section might be published via these lines in a
   domain zone file:

      example.com.   TXT "v=spf1 +mx a:colo.example.com/28 -all"
      smtp-out.example.com. TXT "v=spf1 a -all"

   Since TXT records have multiple uses, beware of other TXT records
   published there for other purposes. They might cause problems with
   size limits (see Section 3.4) and care has to be taken to ensure only
   SPF records are used for SPF processing.

   ADMDs publishing SPF records SHOULD try to keep the number of
   "include" mechanisms and chained "redirect" modifiers to a minimum.
   ADMDs SHOULD also try to minimize the amount of other DNS information
   needed to evaluate a record. Section 4.6.4 and Section 10.1.1
   provide some suggestions on how to achieve this.

3.1. DNS Resource Records

   SPF records MUST be published as a DNS TXT (type 16) Resource Record
   (RR) [RFC1035] only. The character content of the record is encoded
   as [US-ASCII]. Use of alternate DNS RR types was supported in SPF's
   experimental phase, but has been discontinued. See Appendix A of
   [RFC6686] for further information.

3.2. Multiple DNS 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.

3.3. Multiple Strings in a Single DNS record

As defined in [RFC1035] sections 3.3.14 and 3.3, a single text DNS
record can be composed of more than one string. If a published
record contains multiple character-strings, then the record MUST be
treated as if those strings are concatenated together without adding
spaces. For example:

    IN TXT "v=spf1 .... first" "second string..."

MUST be treated as equivalent to:

    IN TXT "v=spf1 .... firstsecond string..."

TXT records containing multiple strings are useful in constructing
records that would exceed the 255-byte maximum length of a character-
string within a single TXT record.

3.4. 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 UDP limit is defined in [RFC1035] section 2.3.4. 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 is under 450 octets, then DNS answers ought to fit in UDP
packets. Records that are too long to fit in a single UDP packet
could be silently ignored by SPF verifiers due to firewall and other
issues that cause DNS over TCP to be less reliable than DNS over UDP.

Note that when computing the sizes for replies to queries of the TXT
format, one has to take into account any other TXT records published
at the domain name. Similarly, the sizes for replies to all queries
related to SPF have to be evaluated to fit in a single UDP packet.

3.5. Wildcard Records

Use of wildcard records for publishing is discouraged and care has to
be taken if they are used. If a zone includes wildcard MX records,
it might 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. Consider the example in [RFC1034], Section
4.3.3. Based on that, we can do the following:

```
    EXAMPLE.COM.           MX      10      A.EXAMPLE.COM
    EXAMPLE.COM.           TXT     "v=spf1 a:A.EXAMPLE.COM -all"

    *.EXAMPLE.COM.         MX      10      A.EXAMPLE.COM
    *.EXAMPLE.COM.         TXT     "v=spf1 a:A.EXAMPLE.COM -all"

    A.EXAMPLE.COM.         A       203.0.113.1
    A.EXAMPLE.COM.         MX      10      A.EXAMPLE.COM
    A.EXAMPLE.COM.         TXT     "v=spf1 a:A.EXAMPLE.COM -all"

    *.A.EXAMPLE.COM.       MX      10      A.EXAMPLE.COM
    *.A.EXAMPLE.COM.       TXT     "v=spf1 a:A.EXAMPLE.COM -all"
```

SPF records have to be listed twice for every name within the zone:
once for the name, and once with a wildcard to cover the tree under
the name, in order to cover all domains in use in outgoing mail.

4. The check_host() Function

   This description is not an API (Application Program Interface)
   definition, but rather a function description used to illustrate the
   algorithm. A compliant SPF implementation MUST do something
   semantically equivalent to this description.

   The check_host() function fetches SPF records, parses them, and
   evaluates them to determine whether a particular host is or is not
   permitted to send mail with a given 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 in all
   cases.

4.1. Arguments

   The check_host() function takes these arguments:

   <ip>   - the IP address of the SMTP client that is emitting the
              mail, either IPv4 or IPv6.

   <domain> - the domain that provides the sought-after authorization
              information; initially, the domain portion of the "MAIL
              FROM" or "HELO" identity.

   <sender> - the "MAIL FROM" or "HELO" identity.

   For recursive evaluations, the domain portion of <sender> might not
   be the same as the <domain> argument when check_host() is initially
   evaluated. In most other cases it will be the same. (See
   Section 5.2 below).

   Note that the <domain> argument might not be a well-formed domain
   name. For example, if the reverse-path was null, then the EHLO/HELO
   domain is used, with its associated problems (see Section 2.1). In
   these cases, check_host() is defined in Section 4.3 to return a
   "none" result.

4.2. Results

   The function check_host() can return one of several results described
   in Section 2.6. Based on the result, the action to be taken is
   determined by the local policies of the receiver. This is discussed
   in Section 8.

4.3. Initial Processing

   If the <domain> is malformed (e.g. label longer than 63 characters,
   zero-length label not at the end, etc.) or is not a fully qualified
   domain name, or if the DNS lookup returns "domain does not exist"
   (RCODE 3), check_host() immediately returns the result "none".
   Properly formed domains are fully qualified email domains as
   described in [RFC5321] Section 2.3.5. Internationalized domain names
   MUST be encoded as A-labels, as described in Section 2.3 of
   [RFC5890].on 2.3 of [RFC5890].

   If the <sender> has no local-part, substitute the string "postmaster"
   for the local-part.

4.4. Record Lookup

   In accordance with how the records are published (see Section 3
   above), a DNS query needs to be made for the <domain> name, querying
   for type TXT only.

   If all DNS lookups that are made return a server failure (RCODE 2),
   or other error (RCODE other than 0 or 3), or time out, then
   check_host() terminates immediately with the result "temperror".
   Alternatively, for a server failure (RCODE 2) result, check_host()
   MAY track failures and treat multiple failures within 24 hours for
   the same domain as "permerror".

   This alternative is intended to shorten the queue time of messages
   that cannot be accepted, by returning a permanent negative completion
   reply code to the client, instead of a transient one. [RFC2308]
   suggests on an algorithm for doing such tracking and handling of
   server failure codes.

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,
   discard records that do not begin with a version section of exactly
   "v=spf1". Note that the version section is terminated either by an
   SP character or the end of the record. A record with a version
   section of "v=spf10" does not match and is discarded.

   If the resultant record set includes no records, check_host()
   produces the "none" result. If the resultant record set includes

more than one record, check_host() produces the "permerror" result.

4.6. Record Evaluation

The check_host() function parses and interprets the SPF record to
find a result for the current test. If there are any syntax errors
anywhere in the record, check_host() returns immediately with the
result "permerror", without further interpretation.

4.6.1. Term Evaluation

There are two types of terms: mechanisms and modifiers. A record
contains an ordered list of these as specified in the following
Augmented Backus-Naur Form (ABNF).

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

directive        = [ qualifier ] mechanism
qualifier        = "+" / "-" / "?" / "~"
mechanism        = ( all / include
                   / A / MX / PTR / IP4 / IP6 / exists )
modifier         = redirect / explanation / unknown-modifier
unknown-modifier = name "=" macro-string
                     ; where name is not any known modifier

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 might be part of
the macro-string.

Terms that do not contain any of "=", ":", or "/" are mechanisms, as
defined in Section 5.

As per the definition of the ABNF notation in [RFC5234], mechanism
and modifier names are case-insensitive.

4.6.2. Mechanisms

Each mechanism is considered in turn from left to right. If there
are no more mechanisms, the result is specified in Section 4.7.

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

If it matches, processing ends and the qualifier value is returned as

the result of that record. If it does not match, processing
continues with the next mechanism. If it returns an exception,
mechanism processing ends and the exception value is returned.

The possible qualifiers, and the results they cause check_host() to
return are as follows:

    "+" pass
    "-" fail
    "~" softfail
    "?" neutral

The qualifier is optional and defaults to "+".

When a mechanism matches and the qualifier is "-", then a "fail"
result is returned and the explanation string is computed as
described in Section 6.2.

The specific mechanisms are described in Section 5.

4.6.3. Modifiers

Modifiers are not mechanisms. They do not return match or not-match.
Instead, they provide additional information. Although modifiers do
not directly affect the evaluation of the record, the "redirect"
modifier has an effect after all the mechanisms have been evaluated.

4.6.4. DNS Lookup Limits

SPF implementations MUST limit the number of mechanisms and modifiers
("terms") that cause any DNS query to at most 10 during SPF
evaluation. Specifically, the "include", "a", "mx", "ptr", and
"exists" mechanisms as well as the "redirect" modifier count against
this limit. The "all", "ip4", and "ip6" mechanisms do not count
against this limit. If this number is exceeded during a check, a
permerror MUST be returned. The "exp" modifier does not count
against this limit because the DNS lookup to fetch the explanation
string occurs after the SPF record evaluation has been completed.

When evaluating the "mx" mechanism, the number of "MX" resource
records queried is included in the overall limit of 10 mechanisms/
modifiers the cause DNS look ups described above. The evaluation of
each "MX" record MUST NOT result in querying more than 10 "A"
resource records. If this limit is exceeded, the "mx" mechanism MUST
produce a "permerror" result.

When evaluating the "ptr" mechanism or the %{p} macro, the number of
"PTR" resource records queried is included in the overall limit of 10

mechanisms/modifiers the cause DNS look ups described above. The
evaluation of each "PTR" record MUST NOT result in querying mor e
than 10 "A" resource records. If this limit is exceeded, all records
other than the first 10 MUST be ignored.

The reason for the disparity is that the set of and contents of the
MX record are under control of the domain owner, while the set of and
contents of PTR records are under control of the owner of the IP
address actually making the connection.

These limits are per mechanism or macro in the record, and are in
addition to the lookup limits specified above.

MTAs or other processors SHOULD 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
authorization SHOULD be "temperror".

4.7. Default Result

If none of the mechanisms match and there is no "redirect" modifier,
then the check_host() returns a result of "neutral", just as if
"?all" were specified as the last directive. If there is a
"redirect" modifier, check_host() proceeds as defined in Section 6.1.

Note that records SHOULD always use either a "redirect" modifier or
an "all" mechanism to explicitly terminate processing. Although the
latter has default (specifically "?all"), it aids debugging efforts
if it is explicitly included.

For example:

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

4.8. Domain Specification

Several of these mechanisms and modifiers have a domain-spec section.
The domain-spec string is subject to macro expansion (see Section 7).
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 (e.g., the 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>. Domain and
domain-spec are syntactically identical after macro expansion.
Domain is an input value for check_host() while domain-spec is
computed by check_host().

Note: Historically, this document has made no provisions for how to
handle domain-specs, or macro-expansions thereof, that are
syntactically invalid per [RFC1035], such as names with empty labels
(e.g., "foo..example.com") or overlong labels (more than 63
characters). Some implementations choose to treat as a no-match
mechanisms, and ignore modifiers with such names, whereas others
return a "permerror" exception. The outcome for an unexpected
domain-spec without macros might even differ from that for an
unexpected target-name after macro expansion.

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 permitted to use the <domain> for
   sending mail.

      a
      mx
      ptr (do not use)
      ip4
      ip6
      exists

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

   If no CIDR prefix length is given in the directive, then <ip> and the
   IP address are compared for equality. (Here, CIDR is Classless
   Inter-Domain Routing, described in [RFC4632].)

   If a CIDR prefix 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, "A" records are fetched; when <ip> is an IPv6
   address, "AAAA" records are fetched. SPF implementations on IPv6
   servers need to handle both "AAAA" and "A" secords, for clients on
   IPv4 mapped IPv6 addresses [RFC4291]. IPv4 <ip> addresses are only
   listed in an SPF record using the "ip4" mechanism.

   Several mechanisms rely on information fetched from the DNS. For
   these DNS queries, except where noted, if the DNS server returns an
   error (RCODE other than 0 or 3) or the query times out, the mechanism
   stops and the topmost check_host() returns "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 a mx -all
```

Mechanisms after "all" will never be tested. Mechanisms listed after
"all" MUST be ignored. Any "redirect" modifier (Section 6.1) MUST be
ignored when there is an "all" mechanism in the record.

5.2. "include"

```
include          = "include"  ":" domain-spec
```

The "include" mechanism triggers a recursive evaluation of
check_host().

1. The domain-spec is expanded as per Section 7.

2. 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().

3. The recursive evaluation returns either match, not match, or an
    error. If it matches, then the appropriate result for the
    include: mechanism is used (e.g. include or +include gives a
    "pass" result and -include gives "fail).

4. If there is no match, the parent check_host() resumes processing
    as per the table below, with the previous value of <domain>
    restored.

In hindsight, the name "include" was poorly chosen. Only the
evaluated result of the referenced SPF record is used, rather than
acting as if the referenced SPF record was literally included in the
first. For example, evaluating a "-all" directive in the referenced
record does not terminate the overall processing and does not
necessarily result in an overall "fail". (Better names for this
mechanism would have been "if-match", "on-match", etc.)

The "include" mechanism 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

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

This 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, does not match, or returns an
exception depends on the result of the recursive evaluation of
check_host():

```
+-------------------------------+-------------------------------+
| A recursive check_host() result | Causes the "include" mechanism |
| of:                           | to:                           |
+-------------------------------+-------------------------------+
| pass                          | match                         |
|                               |                               |
| fail                          | not match                     |
|                               |                               |
| softfail                      | not match                     |
|                               |                               |
| neutral                       | not match                     |
|                               |                               |
| temperror                     | return temperror              |
|                               |                               |
| permerror                     | return permerror              |
|                               |                               |
| none                          | return permerror              |
+-------------------------------+-------------------------------+
```

The "include" mechanism is intended for crossing administrative
boundaries. For example, if example.com and example.org were managed
by the same entity, and if the permitted set of hosts for both
domains was
"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".

With the "include" mechanism an administratively external set of
hosts can be authorized, but determination of sender policy is still
a function of the original domain's SPF record (as determined by the
"all" mechanism in that record). The redirect modifier is more
suitable for consolidating both authorizations and policy into a
common set to be shared within an ADMD. Redirect is much more like a

common code element to be shared among records in a single ADMD. It
is possible to control both authorized hosts and policy for an
arbitrary number of domains from a single record.

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. To prevent Denial of Service
(DoS) attacks, more than 10 MX names MUST NOT be looked up during the
evaluation of an "mx" mechanism. If there are more than 10 MX names
then permerror is returned and the evaluation terminated (see
Section 4.6.4). If any address matches, the mechanism matches.

Note regarding implicit MXs: 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 or AAAA lookup on the <target-name> directly.
This behavior diverges from the legacy "implicit MX" rule, (See
[RFC5321], Section 5. If such behavior is desired, the publisher
will have to specify an "a" directive).

5.5. "ptr" (do not use)

This mechanism tests whether the DNS reverse-mapping for <ip> exists
and correctly points to a domain name within a particular domain.
This mechanism SHOULD NOT be used. See below for discussion.

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

The <ip>'s name is looked up using this procedure:

o Perform a DNS reverse-mapping for <ip>: Look up the corresponding
   PTR record in "in-addr.arpa." if the address is an IPv4 one and in
   "ip6.arpa." if it is an IPv6 address.

o For each record returned, validate the domain name by looking up
   its IP addresses. To prevent DoS attacks, more than 10 PTR names
   MUST NOT be looked up during the evaluation of a "ptr" mechanism
   (see [Section 4.6.4](#)).

o If <ip> is among the returned IP addresses, then that domain name
   is validated.

Check all validated domain names to see if they either match the
<target-name> domain or are a subdomain of the <target-name> domain.
If any do, this mechanism matches. If no validated domain name can
be found, or if none of the validated domain names match or are a
subdomain of the <target-name>, this mechanism fails to match. If a
DNS error occurs while doing the PTR RR lookup, then this mechanism
fails to match. If a DNS error occurs while doing an A RR lookup,
then that domain name is skipped and the search continues.

Pseudocode:

```
sending-domain_names := ptr_lookup(sending-host_IP);
if more than 10 sending-domain_names are found, use at most 10.
for each name in (sending-domain_names) {
  IP_addresses := a_lookup(name);
  if the sending-domain_IP is one of the IP_addresses {
    validated-sending-domain_names += name;
  }
}

for each name in (validated-sending-domain_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 either a subdomain of
a validated domain name or if the <target-name> and a validated
domain name are the same. For example: "mail.example.com" is within
the domain "example.com", but "mail.bad-example.com" is not.

Note: This mechanism is slow, it is not as reliable as other
mechanisms in cases of DNS errors, and it places a large burden on
the .arpa name servers. If used, proper PTR records MUST be in place
for the domain's hosts and the "ptr" mechanism SHOULD be one of the
last mechanisms checked. After many years of SPF deployment

experience it has been concluded it is unnecessary and more reliable
alternatives used instead. It is, however, still in use and part of
the SPF protocol, so compliant check_host() implementations MUST
support it.

5.6. "ip4" and "ip6"

These mechanisms test whether <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
dual-cidr-length = [ ip4-cidr-length ] [ "/" ip6-cidr-length ]

ip4-network      = qnum "." qnum "." qnum "." qnum
qnum             = DIGIT                  ; 0-9
                   / %x31-39 DIGIT        ; 10-99
                   / "1" 2DIGIT           ; 100-199
                   / "2" %x30-34 DIGIT    ; 200-249
                   / "25" %x30-35         ; 250-255
         ; as per conventional dotted quad notation.  e.g., 192.0.2.0
ip6-network      = <as per [RFC 4291], section 2.2>
         ; e.g., 2001:DB8::CD30
```

The <ip> is compared to the given network. If CIDR prefix 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". It is not
permitted to omit parts of the IP address instead of using CIDR
notations. That is, use 192.0.2.0/24 instead of 192.0.2.

5.7. "exists"

This mechanism is used to construct an arbitrary domain 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
permitted.

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

The domain-spec is expanded as per Section 7. The resulting domain
name is used for a DNS A RR lookup (even when the connection type is
IPv6). If any A record is returned, this mechanism matches.

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.

6. Modifier Definitions

   Modifiers are name/value pairs that provide additional information.
   Modifiers always have an "=" separating the name and the value.

   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 two modifiers does not matter.
   These two modifiers MUST NOT appear in a record more than once each.
   If they do, then check_host() exits with a result of "permerror".

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

6.1. redirect: Redirected Query

   The redirect modifier is intended for consolidating both
   authorizations and policy into a common set to be shared within a
   single ADMD. Redirect is like a common code element to be shared
   among records in a single ADMD. It is possible to control both
   authorized hosts and policy for an arbitrary number of domains from a
   single record.

   redirect   = "redirect" "=" domain-spec

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

   The domain-spec portion of the redirect section is expanded as per
   the macro rules in Section 7. 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().

   The result of this new evaluation of check_host() is then considered
   the result of the current evaluation with the exception that if no
   SPF record is found, or if the target-name is malformed, the result
   is a "permerror" rather than "none".

   Note that the newly-queried domain can 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. TXT "v=spf1 redirect=_spf.example.com"
   ny.example.com. TXT "v=spf1 redirect=_spf.example.com"
   sf.example.com. TXT "v=spf1 redirect=_spf.example.com"
_spf.example.com. TXT "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, the 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 mailboxes in domain "A",
especially if domain "B" uses mechanisms involving local-parts. An
"include" directive is generally 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 either a default explanation string or an empty
explanation string MUST be returned.

The domain-spec is macro expanded (see Section 7) and becomes the
<target-name>. The DNS TXT record for the <target-name> is fetched.

If there are any DNS processing errors (any RCODE other than 0), or
if no records are returned, or if more than one record is returned,
or if there are syntax errors in the explanation string, then proceed
as if no exp modifier was given.

The fetched TXT record's strings are concatenated with no spaces, and
then treated as an explain-string, which is macro-expanded. This
final result is the explanation string. Implementations MAY limit
the length of the resulting explanation string to allow for other
protocol constraints and/or reasonable processing limits. Since the
explanation string is intended for an SMTP response and [RFC5321]
Section 2.4 says that responses are in [US-ASCII], the explanation
string MUST be limited to US-ASCII.

Software evaluating check_host() can use this string 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 "%{o} explains: " to the explanation, such as shown in
   [Section 2.6.4](#).

   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:

      "Mail from example.com 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 information, 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, an exp= modifier
   from the <target-name> MUST NOT be used. In contrast, when executing
   a "redirect" modifier, an exp= modifier from the original domain MUST
   NOT be used.

7. Macros

   When evaluating an SPF policy record, certain character sequences are
   intended to be replaced by parameters of the message or of the
   connection. These character sequences are referred to as "macros".

7.1. Formal Specification

   The ABNF description for a macro is as follows:

   domain-spec   = macro-string domain-end
   domain-end   = ("." toplabel ["."]) / macro-expand

   toplabel   = (*alphanum ALPHA *alphanum) /
                     (1*alphanum "-" *(alphanum / "-") alphanum)
   alphanum   = ALPHA / DIGIT

   explain-string = *(macro-string / SP)

   macro-string   = *(macro-expand / macro-literal)
   macro-expand   = ("%{" macro-letter transformers *delimiter "}")
                     / "%%" / "%_" / "%-"
   macro-literal   = %x21-24 / %x26-7E
                     ; visible characters except "%"
   macro-letter   = "s" / "l" / "o" / "d" / "i" / "p" / "h" /
                     "c" / "r" / "t" / "v"
   transformers   = *DIGIT ["r"]
   delimiter   = "." / "-" / "+" / "," / "/" / "_" / "="

   The "toplabel" construction is subject to the LDH rule plus
   additional top-level domain (TLD) restrictions. See Section 2 of
   [RFC3696] for background.

   Some special cases:

   o A literal "%" is expressed by "%%".

   o "%_" expands to a single " " space.

   o "%-" expands to a URL-encoded space, viz., "%20".

7.2. Macro Definitions

   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 domain name of <ip> (do not use)
   v = the string "in-addr" if <ip> is ipv4, or "ip6" if <ip> is ipv6
   h = HELO/EHLO domain
```

<domain>, <sender>, and <ip> are defined in [Section 2.4](#).

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

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

7.3. Notes

A '%' character not followed by a '{', '%', '-', or '_' character is
a syntax error. So:

```
   -exists:%(ir).sbl.spamhaus.example.org
```

is incorrect and will cause check_host() to yield a "permerror".
Instead, the following is legal:

```
   -exists:%{ir}.sbl.spamhaus.example.org
```

Optional transformers are the following:

```
    *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 separated by one or more of the
specified delimiter characters. 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 in
input strings, and so the list of parts might contain empty strings.
Some older implementations of SPF prohibit trailing dots in domain
names, so trailing dots SHOULD NOT be published by domain owners,
although they MUST be accepted by implementations conforming to this
document. Macros can specify delimiter characters that are used
instead of ".".

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 email address with a local-part, an "@" character, and a domain. The "l" macro expands to just the local-part. 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 local-part, the local-part 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 can expand to any of the hexadecimal colon-format addresses specified in [RFC4291], Section 2.2. It is intended for humans to read.

The "p" macro expands to the validated domain name of <ip>. The procedure for finding the validated domain name is defined in Section 5.5. If the <domain> is present in the list of validated domains, it SHOULD be used. Otherwise, if a subdomain of the <domain> is present, it SHOULD be used. Otherwise, any name from the list can be used. If there are no validated domain names or if a DNS error occurs, the string "unknown" is used. This macro SHOULD NOT be used. See Section 5.5 for the discussion about why not.

The "h" macro expands to the parameter that was provided to the SMTP server via the HELO or EHLO SMTP verb. For sessions where that verb was provide more than once, the most recent instance 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 MUA) or if policy restrictions dictate otherwise, the word "unknown" SHOULD be substituted. The domain name can be different from 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

approximate number of seconds since the Epoch (Midnight, January 1,
1970, UTC) at the time of the evaluation. This is the same value as
is returned by the POSIX time() function in most standards-compliant
libraries.

When the result of macro expansion is used in a domain name query, if
the expanded domain name exceeds 253 characters (the maximum length
of a domain name), the left side is truncated to fit, by removing
successive domain labels (and their following dots) until the total
length does not exceed 253 characters.

Uppercased macros expand exactly as their lowercased equivalents, and
are then URL escaped. URL escaping MUST be performed for characters
not in the "unreserved" set, which is defined in [RFC3986].

Note: Care has to be taken by the sending ADMD so that macro
expansion for legitimate email does not exceed the 63-character limit
on DNS labels. The local-part of email addresses, in particular, can
have more than 63 characters between dots.

Note: To minimize DNS lookup resource requirements, it is better if
sending ADMDs avoid using the "s", "l", "o", or "h" macros in
conjunction with any mechanism directive. Although 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.

Note: If no directive processed during the evaluation of check_host()
contains an "s", "l", "o", or "h" macro, then the results of the
evaluation can be cached on the basis of <domain> and <ip> alone for
as long as the shortest Time To Live (TTL) of all the DNS records
involved.

7.4. 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 2001:DB8::CB01.
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.B.C.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._spf.example.com
```

8. Result Handling

   This section provides guidance for operators in response to the
   various possible outputs of check_host() on a message. Terse
   definitions of SPF results are presented in Section 2.6; this section
   provides more detail on each for use in developing local policy for
   message handling.

   Every operating environment is different. There are some receivers
   for whom strict adherence to SPF is appropriate, and definitive
   treatment of messages that are evaluated to be explicity unauthorized
   ("fail" and sometimes "softfail") is the norm. There are others for
   which the "false negative" cases are more of a concern. This concern
   is typically handled by merely recording the result in the header and
   allowing the message to pass on for additional processing. There are
   still others where SPF is one of several inputs to the message
   handling decision. As such, there is no normative requirement for
   message handling in response to any particular result. This section
   is provided to present a complete picture of the likely cause of each
   result, and where available, the experience gained during
   experimental deployment.

   There are essentially two classes of handling choices:

   o Handling within the SMTP session that attempted to deliver the
      message, such as by returning a permanent SMTP error (rejection)
      or temporary SMTP error ("try again later");

   o Permitting the message to pass (a successful SMTP reply code) and
      adding an additional header field that indicates the result
      returned by check_host() and other salient details; this is
      discussed in more detail in Section 9.

8.1. None

   With a "none" result, the SPF verifier has no information at all
   about the authorization or lack thereof of the client to use the
   checked idenity or identities. The check_host() function completed
   without errors but was not able to reach any conclusion.

8.2. Neutral

   A "neutral" result indicates that although a policy for the identity
   was discovered, there is no definite assertion about the (positive or
   negative) about the client.

   A "neutral" result MUST be treated exactly like the "none" result;
   the distinction exists only for informational purposes. Treating

"neutral" more harshly than "none" would discourage domain owners
from testing the use of SPF records (see Section 10.1).

8.3. Pass

A "pass" result means that the client is authorized to inject mail
with the given identity. The domain can now, in the sense of
reputation, be considered responsible for sending the message.
Further policy checks can now proceed with confidence in the
legitimate use of the identity. This is further discussed in
Appendix G.1.

8.4. Fail

A "fail" result is an explicit statement that the client is not
authorized to use the domain in the given identity. Disposition of
SPF fail messages is a matter of local policy. See Appendix G.2 for
considerations on developing local policy.

If the checking software chooses to reject the mail during the SMTP
transaction, then it SHOULD use an SMTP reply code of 550 (see
[RFC5321]) and, if supported, the 5.7.1 enhanced status code (see
[RFC3463]), in addition to an appropriate reply text. The
check_host() function will return either a default explanation string
or one from the domain that published the SPF records (see
Section 6.2). If the information does not originate with the
checking software, it is good to make it clear that the text is
provided by the sender's domain. For example:

    550-5.7.1 SPF MAIL FROM check failed:
    550-5.7.1 The domain example.com explains:
    550 5.7.1 Please see http://www.example.com/mailpolicy.html

If the checking software chooses not to reject the mail during the
SMTP transaction, then it SHOULD add a Received-SPF or
Authentication-Results header field (see Section 9) to communicate
this result to downstream message processors. While this is true for
all SPF results, it is of particular importance for "fail" results
since the message is explicitly not authorized by the domain owner.

8.5. Softfail

A "softfail" result ought to be treated as somewhere between "fail"
and "neutral"/"none". The domain owner believes the host is not
authorized but is not willing to make a strong policy statement.
Receiving software SHOULD NOT reject the message based solely on this
result, but MAY subject the message to closer scrutiny than normal.

The domain owner wants to discourage the use of this host and thus
desires limited feedback when a "softfail" result occurs. For
example, the recipient's Mail User Agent (MUA) could highlight the
"softfail" status, or the receiving MTA could give the sender a
message using greylisting, [RFC6647], with a note the first time the
message is received, but accept it on a later attempt based on
receiver policy.

8.6. Temperror

A "temperror" result means the SPF verifier encountered a transient
(generally DNS) error while 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 SHOULD use an SMTP reply code of 451 and, if supported, the
4.4.3 enhanced status code. These errors can be caused by problems
in either the sender's or receiver's DNS software.

8.7. Permerror

A "permerror" result means the domain's published records could not
be correctly interpreted. This signals an error condition that
definitely requires manual intervention to be resolved. If the
message is rejected during the SMTP transaction for this reason, the
software SHOULD use an SMTP reply code of 550 and, if supported, the
5.5.2 enhanced status code. Be aware that if the domain owner uses
macros (Section 7), it is possible that this result is due to the
checked identities having an unexpected format. It is also possible
that this result is generated by certain SPF clients due to the input
arguments having an unexpected format; see Section 4.8.

9. Recording The Result

   To provide downstream agents, such as MUAs, with the information they
   might need in terms of evaluating or representing the apparent safety
   of the message content, it is RECOMMENDED that SMTP receivers record
   the result of SPF processing in the message header. For operators
   that choose to record SPF results in the header of the message for
   processing by internal filters or MUAs, two methods are presented.
   Section 9.1 defines the Received-SPF field, which is the results
   field originally defined for SPF use.   Section 9.2 discusses
   Authentication-Results [RFC5451] which was specified more recently
   and is designed for use by SPF and other authentication methods.

   Both are in common use, and hence both are included here. However,
   it is important to note that they were designed to serve slightly
   different purposes. Received-SPF is intended to include enough
   forensic information to enable reconstruction of the SPF evaluation
   of the message, while Authentication-Results is designed only to
   relay the result itself and related output details of likely use to
   end users (e.g., what property of the message was actually
   authenticated and what it contained), leaving forensic work to the
   purview of system logs and the Received field contents. Also,
   Received-SPF relies on compliance of agents within the receiving ADMD
   to adhere to the header field ordering rules of [RFC5321] and
   [RFC5322], while Authentication-Results includes some provisions to
   protect against non-compliant implementations.

   An operator could choose to use both to serve different downstream
   agents. In such cases, care needs to be taken to ensure both fields
   are conveying the same details, or unexpected results can occur.

9.1. The Received-SPF Header Field

   The Received-SPF header field is a trace field (see [RFC5322] Section
   3.6.7) and SHOULD be prepended to the existing header, above the
   Received: field that is generated by the SMTP receiver. It MUST
   appear above all other Received-SPF fields in the message. The
   header field has the following format:

```
   header-field      = "Received-SPF:" [CFWS] result FWS [comment FWS]
                       [ key-value-list ] CRLF

   result            = "pass" / "fail" / "softfail" / "neutral" /
                       "none" / "temperror" / "permerror"

   key-value-list    = key-value-pair *( ";" [CFWS] key-value-pair )
                       [";"]

   key-value-pair    = key [CFWS] "=" ( dot-atom / quoted-string )

   key               = "client-ip" / "envelope-from" / "helo" /
                       "problem" / "receiver" / "identity" /
                        "mechanism" / name

   identity          = "mailfrom"   ; for the "MAIL FROM" identity
                       / "helo"     ; for the "HELO" identity
                       / name       ; other identities

   dot-atom          = <unquoted word as per [RFC5322]>
   quoted-string     = <quoted string as per [RFC5322]>
   comment           = <comment string as per [RFC5322]>
   CFWS              = <comment or folding white space as per [RFC5322]>
   FWS               = <folding white space as per [RFC5322]>
   CRLF              = <standard end-of-line token as per [RFC2532]>
```

The header field SHOULD include a "(...)" style comment after the
result, conveying supporting information for the result, such as
<ip>, <sender>, and <domain>.

The following key-value pairs are designed for later machine parsing.
SPF verifiers SHOULD give enough information so that the SPF results
can be verified. That is, at least "client-ip", "helo", and, if the
"MAIL FROM" identity was checked, "envelope-from".

```
   client-ip      the IP address of the SMTP client

   envelope-from  the envelope sender mailbox

   helo           the host name given in the HELO or EHLO command

   mechanism      the mechanism that matched (if no mechanisms matched,
                  substitute the word "default")

   problem        if an error was returned, details about the error
```

receiver   the host name of the SPF verifier

identity   the identity that was checked; see the <identity> ABNF
                rule

Other keys MAY be defined by SPF verifiers.

SPF verifiers MUST make sure that the Received-SPF header field does
not contain invalid characters, is not excessively long (See
[RFC5322] Section 2.1.1), and does not contain malicious data that
has been provided by the sender.

Examples of various header field styles that could be generated are
the following:

```
Received-SPF: pass (mybox.example.org: domain of
 myname@example.com designates 192.0.2.1 as permitted sender)
    receiver=mybox.example.org; client-ip=192.0.2.1;
    envelope-from="myname@example.com"; helo=foo.example.com;

Received-SPF: fail (mybox.example.org: domain of
                 myname@example.com does not designate
                 192.0.2.1 as permitted sender)
                 identity=mailfrom; client-ip=192.0.2.1;
                 envelope-from="myname@example.com";

Received-SPF: pass (mybox.example.org: domain of
 myname@example.com designates 192.0.2.1 as permitted sender)
    receiver=mybox.example.org; client-ip=192.0.2.1;
    mechanism=ip4:192.0.2.1; envelope-from="myname@example.com";
    helo=foo.example.com;
```

9.2. SPF Results in the Authentication-Results Header Field

As mentioned in Section 9, the Authentication-Results header field is
designed to communicate lists of tests a border MTA did and their
results. The specified elements of the field provide less
information than the Received-SPF field:

```
Authentication-Results: myhost.example.org; spf=pass
  smtp.mailfrom=example.net

Received-SPF: pass (myhost.example.org: domain of
 myname@example.com designates 192.0.2.1 as permitted sender)
    receiver=mybox.example.org; client-ip=192.0.2.1;
    envelope-from="myname@example.com"; helo=foo.example.com;
```

It is, however, possible to add CFWS in the "reason" part of an

Authentication-Results header field and provide the equivalent
information, if desired.

As an example, an expanded Authentication-Results header field might
look like (for a "MAIL FROM" check in this example):

Authentication-Results: myhost.example.org; spf=pass
  reason="client-ip=192.0.2.1; smtp.helo=foo.example.com"
  smtp.mailfrom=user@example.net

10. Effects on Infrastructure

   This section outlines the major implications that adoption of this
   document will have on various entities involved in Internet email.
   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, or a "best practices" document, and it is not
   a comprehensive list of what such entities SHOULD do in light of this
   document.

   This section provides operational advice and instruction only. It is
   non-normative.

   [RFC5598] describes the Internet email architecture. This section is
   organized based on the different segments of the architecture.

10.1. Sending Domains

   Originating ADMDs (ADministrative Management Domains - [RFC5598]
   Section 2.2.1 and Section 2.3) that wish to be compliant with this
   specification will need to determine the list of relays ([RFC5598]
   Section 2.2.2) that they allow to use their domain name in the "HELO"
   and "MAIL FROM" identities when relaying to other ADMDs. 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.

10.1.1. DNS Resource Considerations

   Minimizing the DNS resources required for SPF lookups can be done by
   choosing directives that require less DNS information and by placing
   lower-cost mechanisms earlier in the SPF record.

```
          +----------+--------+-----------------+
          | term     | cost   | limit           |
          +----------+--------+-----------------+
          | ip4/ip6  | 0      | -               |
          | a        | 1      | 10              |
          | include  | 1      | 10              |
          | redirect | 1      | 10              |
          | exists   | 1      | 10              |
          | mx       | 1 + N* | 10 and N* <= 10 |
          | ptr/%{p} | 1 + N* | 10 and N* <= 10 |
          | all      | 0      | -               |
          +----------+--------+-----------------+
           * N is the number of RRs found during each term evaluation
```

   Section 4.6.4 specifies the limits receivers have to use. It is

essential to publish records that do not exceed these requirements.
It is also required to carefully weight the cost and the
maintainability of licit solutions.

For example, consider a domain set up as follows:

```
example.com.     IN MX   10 mx.example.com.
                 IN MX   20 mx2.example.com.
mx.example.com.  IN A    192.0.2.1
mx2.example.com. IN A    192.0.2.129
```

Assume the administrative point is to authorize (pass) mx and mx2
while failing every other host. Compare the following solutions:

```
Best record:
   example.com.   IN TXT  "v=spf1 ip4:192.0.2.1 ip4:192.0.2.129 -all"
```

```
Good record:
   $ORIGIN example.com.
   @              IN TXT  "v=spf1 a:authorized-spf.example.com -all"
   authorized-spf IN A    192.0.2.1
                  IN A    192.0.2.129
```

```
Expensive record:
   example.com.   IN TXT  "v=spf1 mx:example.com -all"
```

```
Wasteful, bad record:
   example.com.   IN TXT  "v=spf1 ip4:192.0.2.0/24 mx -all"
```

10.1.2. Administrator's Considerations

There might be administrative considerations: using "a" over "ip4" or
"ip6" allows hosts to be renumbered easily. Using "mx" over "a"
allows the set of mail hosts to be changed easily. Unless such
changes are common, it is better to use the less resource intensive
mechanisms like "ip4" and "ip6" over "a" or "a" or "mx".

In some specific cases, standard advice on record content is
appropriate. Publishing SPF records for domains that send no mail is
a well established best practice. The record for a domain that sends
no mail is:

```
www.example.com.   IN TXT  "v=spf1 -all"
```

Publishing SPF records for individual hosts is also best practice.

The hostname is generally the identity used in the 5321.HELO/.EHLO
command. In the case of messages with a null 5321.MailFrom, this is
used as the domain for 5321.MailFrom SPF checks, in addition to being
used in 5321.HELO/.EHLO based SPF checks. The standard SPF record
for an individual host that is involved in mail processing is:

    relay.example.com.   IN TXT "v=spf1 a -all"

Validating correct deployment is difficult. [RFC6652] describes one
mechanism for soliciting feedback on SPF failures. Another
suggestion can be found in Appendix C.

Regardless of the method used, understanding the ADMD's outbound mail
architecture is essential to effective deployment.

10.1.3. Bounces

As explained in Section 1.1.3, [RFC5321] allows the reverse-path to
be null, which is typical of some Delivery Status Notification
[RFC3464], commonly called email bounces. In this case the only
entity available for performing an SPF check is the "HELO" identity
defined in Section 1.1.4. SPF functionality is enhanced by
administrators ensuring this identity is set correctly and has an
appropriate SPF record. It is normal to have the HELO identity set
to hostname instead of domain. Zone file generation for significant
numbers of hosts can be consolidated using the redirect modifier and
scripted for initial deployment. Specific deployment advice is given
above in Section 10.1.2.

10.2. Receivers

SPF results can be used in combination with other methods to
determine the final local disposition (either positive or negative of
a message. It can also be considered dispositive on its own.

An attempt to have one organization (sender) direct the email
handling policies of another (receiver) is inherently challenging and
often controversial. As stated elsewhere in this document, there is
no normative requirement for specific handling of a message based on
any SPF result. The information presented in Section 8 and in
Appendix G is offered for receiver consideration when forming local
handling policies.

The primary considerations are that SPF might return "pass" for mail
that is ultimately harmful (e.g., spammers that arrange for SPF to
pass using nonsense domain names, or virus or spam outbreaks from
within trusted sources), and might also return "fail" for mail that
is ultimately legitimate (e.g., legitimate mail that has traversed a

   mail alias). It is important take both of these cases under
   consideration when establishing local handling policy.

10.3. Mediators

   Broadly speaking, there are two types of mediating ADMDs that can
   affect SPF deployment of other ADMDs: mailing lists (see [RFC5598]
   Section 5.3) and ReSenders ([RFC5598] Section 5.2).

10.3.1. Mailing Lists

   Mailing lists have to be aware of how they re-inject mail that is
   sent to the list. Mailing lists MUST comply with the requirements in
   [RFC5321], Section 3.10, and [RFC1123], Section 5.3.6, that say that
   the reverse-path MUST be changed to be the mailbox of a person or
   other entity who administers the list. Whereas the reasons for
   changing the reverse-path are many and long-standing, SPF adds
   enforcement to this requirement.

   In practice, almost all mailing list software in use already complies
   with this requirement. Mailing lists that do not comply might
   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.

10.3.2. Forwarding Services and Aliases

   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 "MAIL
   FROM" of a message when re-injecting it for delivery to the external
   mailbox. [RFC1123] and [RFC5321] describe this action as an "alias"
   rather than a "mail list". 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.

   Appendix D provides some operational suggestions to adapt these
   services to an SPF-aware environment.

11. Security Considerations

11.1. Processing Limits

   As with most aspects of email, there are a number of ways that
   malicious parties could use the protocol as an avenue for a
   Denial-of-Service (DoS) attack. The processing limits outlined in
   Section 4.6.4 are designed to prevent attacks such as the following:

   o A malicious party could create an SPF record with many references
      to a victim's domain and send many emails to different SPF
      verifiers; those SPF verifiers would then create a DoS attack. In
      effect, the SPF verifiers are being used to amplify the attacker's
      bandwidth by using fewer bytes in the SMTP session than are used
      by the DNS queries. Using SPF clients also allows the attacker to
      hide the true source of the attack.

   o Whereas implementations of check_host() are supposed to limit the
      number of DNS lookups, malicious domains could publish records
      that exceed these limits in an attempt to waste computation effort
      at their targets when they send them mail. Malicious domains
      could also design SPF records that cause particular
      implementations to use excessive memory or CPU usage, or to
      trigger bugs.

   o Malicious parties could send a large volume of 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.

   Of these, the case of a third party referenced in the SPF record is
   the easiest for a DoS attack to effectively exploit. As a result,
   limits that might seem reasonable for an individual mail server can
   still allow an unreasonable amount of bandwidth amplification.
   Therefore, the processing limits need to be quite low.

11.2. SPF-Authorized Email May Contain Other False Identities

   Do not construe the "MAIL FROM" and "HELO" identity authorizations to
   provide more assurance than they do. It is entirely possible for a
   malicious sender to inject a message using his own domain in the
   identities used by SPF, to have that domain's SPF record authorize
   the sending host, and yet the message can easily list other
   identities in its header. Unless the user or the MUA takes care to
   note that the authorized identity does not match the other more
   commonly-presented identities (such as the From: header field), the
   user might be lulled into a false sense of security.

11.3. Spoofed DNS and IP Data

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

   o 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 DNS weaknesses.

   o The client IP address, <ip>, is assumed to be correct. In a
      modern, correctly configured system the risk of this not being
      true is nil.

11.4. Cross-User Forgery

   By definition, SPF policies just map domain names to sets of
   authorized MTAs, not whole email addresses to sets of authorized
   users. Although the "l" macro (Section 7) provides a limited way to
   define individual sets of authorized MTAs for specific email
   addresses, it is generally impossible to verify, through SPF, the use
   of specific email addresses by individual users of the same MTA.

   It is up to mail services and their MTAs to directly prevent
   cross-user forgery: based on SMTP AUTH ([RFC4954]), users have to be
   restricted to using only those email addresses that are actually
   under their control (see [RFC6409], Section 6.1). Another means to
   verify the identity of individual users is message cryptography such
   as PGP ([RFC4880]) or S/MIME ([RFC5751]).

11.5. Untrusted Information Sources

   An SPF compliant receiver gathers information from the SMTP commands
   it receives and from the published DNS records of the sending domain
   holder, (e.g., "HELO" domain name, the "MAIL FROM" address from the
   envelope, and SPF DNS records published by the domain holder).

11.5.1. Recorded Results

   This information, passed to the receiver in the Received-SPF: or
   Authentication-Results: trace fields, may be returned to the client
   MTA as an SMTP rejection message. If such an SMTP rejection message
   is generated, the information from the trace fields has to be checked
   for such problems as invalid characters and excessively long lines.

11.5.2. **External Explanations**

   When the authorization check fails, an explanation string could 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, in general, not the
   receiver. The explanation can contain malicious URLs, or it might be
   offensive or misleading.

   Explanations returned to sender domains due to "exp" modifiers,
   ([Section 6.2]), were generated by the sender policy published by the
   domain holders themselves. As long as messages are only returned
   with non-delivery notification ([[RFC3464]]) to domains publishing the
   explanation strings from their own DNS SPF records, the only affected
   parties are the original publishers of the domain's SPF records.

   In practice, such non-delivery notifications can be misdirected, such
   as when an MTA accepts an email and only later generates the
   notification to a forged address, or when an email forwarder does not
   direct the bounce back to the original sender.

11.5.3. **Macro Expansion**

   Macros ([Section 7]) allow senders to inject arbitrary text (any non-
   null [[US-ASCII]] character) into receiver DNS queries. It is necesary
   to be prepared for hostile or unexpected content.

11.6. **Privacy Exposure**

   Checking SPF records causes DNS queries to be sent to the domain
   owner. These DNS queries, especially if they are caused by the
   "exists" mechanism, can contain information about who is sending
   email and likely to which MTA the email is being sent. This can
   introduce some privacy concerns, which are more or less of an issue
   depending on local laws and the relationship between the domain owner
   and the person sending the email.

11.7. **Delivering Mail Producing a 'Fail' Result**

   Operators that choose to deliver mail for which SPF produces a "fail"
   result need to understand that they are admitting content that is
   explicitly not authorized by the purported sender. While there are
   known failure modes that can be considered "false negatives", the
   distinct choice to admit those messages increases end-user exposure
   to likely harm. This is especially true for domains belonging to
   known good actors that are typically well-behaved; unauthorized mail
   from those sources might well be subjected to much higher skepticism
   and content analysis.

SPF does not, however, include the capacity for identifying good
actors from bad ones, nor does it handle the concept of known actors
versus unknown ones. Those notions are out of scope for this
specification.

12. Contributors and Acknowledgements

   This document is largely based on the work of Meng Weng Wong, Mark
   Lentczner, and Wayne Schlitt. Although, as this section
   acknowledges, many people have contributed to this document, a very
   large portion of the writing and editing are due to Meng, Mark, and
   Wayne.

   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 further 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 the concept of macros to the
   specification, multiplying the expressiveness of the language and
   making per-user and per-IP lookups possible.

   The authors of both this document and [RFC4408] would also like to
   thank the literally hundreds of individuals who have participated in
   the development of this design. They are far too numerous to name,
   but they include the following:

      The participants in the SPFbis working group.
      The folks on the spf-discuss mailing list.
      The folks on the SPAM-L mailing list.
      The folks on the IRTF ASRG mailing list.
      The folks on the IETF MARID mailing list.
      The folks on #perl.

13. IANA Considerations

13.1. The SPF DNS Record Type

   Per [RFC4408], the IANA assigned the Resource Record Type and Qtype
   from the DNS Parameters Registry for the SPF RR type with code 99.
   The format of this type is identical to the TXT RR [RFC1035]. The
   character content of the record is encoded as [US-ASCII]. Use of
   this record type is obsolete for SPF Version 1.

   IANA is requested to add an annotation to the SPF RRTYPE saying
   "(OBSOLETE - use TXT)" in the DNS Parameters registry.

   [NOTE TO RFC EDITOR: (to be changed to " ... has added ..." upon
   publication)]

13.2. The Received-SPF Mail Header Field

   Per [RFC3864], the "Received-SPF:" header field is added to the IANA
   Permanent Message Header Field Registry. The following is the
   registration template:

      Header field name: Received-SPF
      Applicable protocol: mail ([RFC5322])
      Status: Standards Track
      Author/Change controller: IETF
      Specification document(s): RFC XXXX
      [NOTE TO RFC EDITOR: (this document)]

13.3. SPF Modifier Registration

   [RFC6652] created a new SPF Modifier Registration. IANA is requested
   to change the reference for the exp and redirect modifiers from
   [RFC4408] to this document. Their status should not be changed.

14. References

14.1. Normative References

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

   [RFC1123] Braden, R., "Requirements for Internet Hosts - Application
              and Support", STD 3, RFC 1123, October 1989.

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

   [RFC3463] Vaudreuil, G., "Enhanced Mail System Status Codes",
              RFC 3463, January 2003.

   [RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration
              Procedures for Message Header Fields", BCP 90, RFC 3864,
              September 2004.

   [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
              Resource Identifier (URI): Generic Syntax", STD 66,
              RFC 3986, January 2005.

   [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing
              Architecture", RFC 4291, February 2006.

   [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax
              Specifications: ABNF", STD 68, RFC 5234, January 2008.

   [RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321,
              October 2008.

   [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322,
              October 2008.

   [RFC5451] Kucherawy, M., "Message Header Field for Indicating
              Message Authentication Status", RFC 5451, April 2009.

   [RFC5598] Crocker, D., "Internet Mail Architecture", RFC 5598,
              July 2009.

   [RFC5890] Klensin, J., "Internationalized Domain Names for
              Applications (IDNA): Definitions and Document Framework",
              RFC 5890, August 2010.

   [US-ASCII]
              American National Standards Institute (formerly United

States of America Standards Institute), "USA Code for
Information Interchange, X3.4", 1968.

ANSI X3.4-1968 has been replaced by newer versions with
slight modifications, but the 1968 version remains
definitive for the Internet.

14.2. Informative References

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

           Work In Progress

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

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

[RFC1983] Malkin, G., "Internet Users' Glossary", RFC 1983,
           August 1996.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS
           NCACHE)", RFC 2308, March 1998.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for
           specifying the location of services (DNS SRV)", RFC 2782,
           February 2000.

[RFC3464] Moore, K. and G. Vaudreuil, "An Extensible Message Format
           for Delivery Status Notifications", RFC 3464,
           January 2003.

[RFC3696] Klensin, J., "Application Techniques for Checking and
           Transformation of Names", RFC 3696, February 2004.

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

[RFC3834] Moore, K., "Recommendations for Automatic Responses to
           Electronic Mail", RFC 3834, August 2004.

[RFC4408] Wong, M. and W. Schlitt, "Sender Policy Framework (SPF)
           for Authorizing Use of Domains in E-Mail, Version 1",
           RFC 4408, April 2006.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing
           (CIDR): The Internet Address Assignment and Aggregation
           Plan", BCP 122, RFC 4632, August 2006.

   [RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R.
              Thayer, "OpenPGP Message Format", RFC 4880, November 2007.

   [RFC4954] Siemborski, R. and A. Melnikov, "SMTP Service Extension
              for Authentication", RFC 4954, July 2007.

   [RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet
              Mail Extensions (S/MIME) Version 3.2 Message
              Specification", RFC 5751, January 2010.

   [RFC6409] Gellens, R. and J. Klensin, "Message Submission for Mail",
              STD 72, RFC 6409, November 2011.

   [RFC6647] Kucherawy, M. and D. Crocker, "Email Greylisting: An
              Applicability Statement for SMTP", RFC 6647, June 2012.

   [RFC6652] Kitterman, S., "Sender Policy Framework (SPF)
              Authentication Failure Reporting Using the Abuse Reporting
              Format", RFC 6652, June 2012.

   [RFC6686] Kucherawy, M., "Resolution of the Sender Policy Framework
              (SPF) and Sender ID Experiments", RFC 6686, July 2012.

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

              Work In Progress

   [Vixie]   Vixie, P., "Repudiating MAIL FROM", 2002.

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 [RFC5234] 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        = [ qualifier ] mechanism
   qualifier        = "+" / "-" / "?" / "~"
   mechanism        = ( all / include
                        / A / MX / PTR / IP4 / IP6 / exists )

   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

   modifier         = redirect / explanation / unknown-modifier
   redirect         = "redirect" "=" domain-spec
   explanation      = "exp" "=" domain-spec
   unknown-modifier = name "=" macro-string
                        ; where name is not any known modifier

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

   ip4-network      = qnum "." qnum "." qnum "." qnum
   qnum             = DIGIT                 ; 0-9
                        / %x31-39 DIGIT       ; 10-99
                        / "1" 2DIGIT          ; 100-199
                        / "2" %x30-34 DIGIT   ; 200-249
                        / "25" %x30-35        ; 250-255
             ; conventional dotted quad notation.  e.g., 192.0.2.0
   ip6-network      = <as per [RFC 4291], section 2.2>
             ; e.g., 2001:DB8::CD30
```

```
   domain-spec       = macro-string domain-end
   domain-end        = ( "." toplabel [ "." ] ) / macro-expand

   toplabel          = ( *alphanum ALPHA *alphanum ) /
                       ( 1*alphanum "-" *( alphanum / "-" ) alphanum )
                       ; LDH rule plus additional TLD restrictions
                       ; (see [RFC3696], Section 2 for background)
   alphanum          = ALPHA / DIGIT

   explain-string    = *( macro-string / SP )

   macro-string      = *( macro-expand / macro-literal )
   macro-expand      = ( "%{" macro-letter transformers *delimiter "}" )
                       / "%%" / "%_" / "%-"
   macro-literal     = %x21-24 / %x26-7E
                       ; visible characters except "%"
   macro-letter      = "s" / "l" / "o" / "d" / "i" / "p" / "h" /
                       "c" / "r" / "t" / "v"
   transformers      = *DIGIT [ "r" ]
   delimiter         = "." / "-" / "+" / "," / "/" / "_" / "="

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

   header-field      = "Received-SPF:" [CFWS] result FWS [comment FWS]
                       [ key-value-list ] CRLF

   result            = "pass" / "fail" / "softfail" / "neutral" /
                       "none" / "temperror" / "permerror"

   key-value-list    = key-value-pair *( ";" [CFWS] key-value-pair )
                       [";"]

   key-value-pair    = key [CFWS] "=" ( dot-atom / quoted-string )

   key               = "client-ip" / "envelope-from" / "helo" /
                       "problem" / "receiver" / "identity" /
                        "mechanism" / name

   identity          = "mailfrom"   ; for the "MAIL FROM" identity
                       / "helo"     ; for the "HELO" identity
                       / name       ; other identities

   ALPHA             = <A-Z / a-z as per [RFC5234]>
   DIGIT             = <0-9 as per [RFC5234]>
   SP                = <space character as per [RFC5234]>
   domain            = <fully qualified domain as per [RFC5321]>
   dot-atom          = <unquoted word as per [RFC5322]>
   quoted-string     = <quoted string as per [RFC5322]>
```

```
comment          = <comment string as per [RFC5322]>
CFWS             = <comment or folding white space as per [RFC5322]>
FWS              = <folding white space as per [RFC5322]>
CRLF             = <standard end-of-line token as per [RFC5322]>
authserv-id      = <authserv-id per [RFC5451]>
reasonspec       = <reason per [RFC5451]>
```

[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 DNS is valid and is in
       example.com)
   --  sending host 192.0.2.140 fails (reverse DNS 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's 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 be updated when the mail setup
changes. These domains' records never have to change.

[B.3](). DNSBL 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}
```

[B.4](). Multiple Requirements Example

Say that your sender policy requires both that the IP address is
within a certain range and that the reverse DNS for the IP matches.
This can be done several ways, including the following:

```
example.com.              SPF  ( "v=spf1 "
                                 "-include:ip4._spf.%{d} "
                                 "-include:ptr._spf.%{d} "
                                 "+all" )
ip4._spf.example.com.  SPF  "v=spf1 -ip4:192.0.2.0/24 +all"
ptr._spf.example.com.  SPF  "v=spf1 -ptr +all"
```

This example shows how the "-include" mechanism can be useful, how an
SPF record that ends in "+all" can be very restrictive, and the use
of De Morgan's Law.

Appendix C. Further Testing Advice

   Another approach that can be helpful to publish records that include
   a "tracking exists:" mechanism. By looking at the name server logs,
   a rough list can then be generated. For example:

      v=spf1 exists:_h.%{h}._l.%{l}._o.%{o}._i.%{i}._spf.%{d} ?all

[Appendix D](). **Updating Mail Forwarders**

There are three places that techniques can be used to ameliorate this problem.

1. The beginning, when email is first sent (Originating ADMDs).

    *   "Neutral" results could be given for IP addresses that might be forwarders, instead of "fail" results based on a list of known reliable forwarders. For example:

        "v=spf1 mx ?exists:%{ir}.whitlist.example.org -all"

        This would cause a lookup on an DNS white list (DNSWL) and cause a result of "fail" only for email not either coming from the domain's mx host(s) (SPF pass) or white listed sources (SPF neutral). This, in effect, outsources an element of sender policy to the maintainer of the whitelist.

    *   The "MAIL FROM" identity could have additional information in the local-part that cryptographically identifies the mail as coming from an authorized source. In this case, such an SPF record could be used:

        "v=spf1 mx exists:%{l}._spf_verify.%{d} -all"

        Then, a specialized DNS server can be set up to serve the _spf_verify subdomain that validates the local-part. Although this requires an extra DNS lookup, this happens only when the email would otherwise be rejected as not coming from a known good source.
        Note that due to the 63-character limit for domain labels, this approach only works reliably if the local-part signature scheme is guaranteed either to only produce local-parts with a maximum of 63 characters or to gracefully handle truncated local-parts.

    *   Similarly, a specialized DNS server could be set up that will rate-limit the email coming from unexpected IP addresses.

        "v=spf1 mx exists:%{ir}._spf_rate.%{d} -all"

    *   SPF allows the creation of per-user policies for special cases. For example, the following SPF record and appropriate wildcard DNS records can be used:

        "v=spf1 mx redirect=%{l1r+}._at_.%{o}._spf.%{d}"

2. The middle, when email is forwarded (Mediating ADMDs).

    *   Forwarding services can solve the problem by rewriting the
        "MAIL FROM" to be in their own domain. This means mail
        rejected from the external mailbox will have to be forwarded
        back to the original sender 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.

    *   Several popular MTAs can be forced from "alias" semantics to
        "mailing list" semantics by configuring an additional alias
        with "owner-" prepended to the original alias name (e.g., an
        alias of "friends: george@example.com, fred@example.org" would
        need another alias of the form "owner-friends: localowner").

    *   Forwarding servers could reject mail that would "fail" SPF if
        forwarded using an SMTP reply code of 551, User not local,
        (see [RFC5321] section 3.4) to communicate the correct target
        address to resend the mail to.

3. The end, when email is received (Receiving ADMDs).

    *   If the owner of the external mailbox wishes to trust the
        forwarding service, he can direct the external mailbox's MTA
        to skip SPF tests when the client host belongs to the
        forwarding service.

    *   Tests against other identities, such as the "HELO" identity,
        MAY be used to override a failed test against the "MAIL FROM"
        identity.

    *   For larger domains, it might 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, whitelists
        of generally-recognized forwarding services could be employed.

Appendix E. **Mail Services**

   MSPs (Mail Service Providers - [RFC5598] Section 2.3) that offer mail
   services to third-party domains, such as sending of bulk mail, might
   want to adjust their configurations in light of the authorization
   check described in this document. If the domain part of the "MAIL
   FROM" identity used for such email uses the domain of one of the MSPs
   domain, then the provider needs only to ensure that its sending host
   is authorized by its own SPF record, if any.

   If the "MAIL FROM" identity does not use the MSP's domain, then extra
   care has to be taken. The SPF record format has several options for
   the third-party domain to authorize the service provider's MTAs to
   send mail on its behalf. For MSPs, such as ISPs, that have a wide
   variety of customers using the same MTA, steps are required to
   mitiate the risk of cross-customer forgery (see Section 11.4).

Appendix F. MTA Relays

   Relays are described in [RFC5598] Section 2.2.2. The authorization
   check generally precludes the use of arbitrary MTA relays between
   sender and receiver of an email message.

   Within an organization, MTA relays can be effectively deployed.
   However, for purposes of this document, such relays are effectively
   transparent. The SPF authorization check is a check between border
   MTAs of different ADMDs.

   For mail senders, this means that published SPF records have to
   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 relaying.

   The receiving ADMD will generally want to perform the authorization
   check at the boundary MTAs, including all secondary MXs. Internal
   MTAs (including MTAs that might serve both as boundary MTAs and
   internal relays from secondary MXs when they are processing the
   relayed mail stream) then do not perform the authorization test. To
   perform the authorization test other than at the boundary, the host
   that first transferred the message to the receiving ADMD have to be
   determined, which can be difficult to extract from the message header
   because (a) header fields can be forged or malformed, and (b) there's
   no standard way to encode that information such that it can be
   reliably extracted. Testing other than at the boundary is likely to
   produce unreliable results.

Appendix G. Local Policy Considerations

   SPF results can be used in combination with other methods to
   determine the final local disposition (either positive or negative of
   a message. It can also be considered dispositive on its own.

G.1. Policy For SPF Pass

   SPF pass results can be used in combination with "white lists" of
   known "good" domains to bypass some or all additional pre-delivery
   email checks. Exactly which checks and how to determine appropriate
   white list entries has to be based on local conditions and
   requirements.

G.2. Policy For SPF Fail

   SPF fail results can be used to reject messages during the SMTP
   transaction based on either "MAIL FROM" or "HELO" identity results.
   This reduces resource requirements for various content filtering
   methods and conserves bandwidth since rejection can be done before
   the SMTP content is transferred. It also gives immediate feedback to
   the sender who might then be able to resolve the issue. Due to some
   of the issues described above in this section (Section 10), SPF based
   rejection does present some risk of rejecting legitimate email when
   rejecting based on "MAIL FROM" results.

   SPF fail results can alternately be used as one input into a larger
   set of evaluations which might, based on a combination with other
   evaluation techniques, result in the email being marked negatively in
   some way (this might be via delivery to a special spam folder,
   modifying subject lines, or other locally determined means).
   Developing the details of such an approach have to be based on local
   conditions and requirements. Using SPF results in this way does not
   have the advantages of resource conservation and immediate feedback
   to the sender associated with SMTP rejection, but could produce fewer
   undesirable rejections in a well designed system. Such an approach
   might result in email that was not authorized by the sending ADMD
   being unknowingly delivered to end users.

   Either general approach can be used as they both leave a clear
   disposition of emails. They are either delivered in some manner or
   the sender is notified of the failure. Other dispositions such as
   "dropping" or deleting email after acceptance are inappropriate
   because they leave uncertainty and reduce the overall reliabilility
   and utility of email across the Internet.

G.3. **Policy For SPF Permerror**

   The "permerror" result (see Section 2.6.7) indicates the SPF
   processing module at the receiver determined that the retrieved SPF
   policy record could not be interpreted. This gives no true
   indication about the authorized use of the data found in the
   envelope.

   As with all results, implementers have a choice to make regarding
   what to do with a message that yields this result. SMTP allows only
   a few basic options.

   Rejection of the message is an option, in that it is the one thing a
   receiver can do to draw attention to the difficulty encountered while
   protecting itself from messages that do not have a definite SPF
   result of some kind. However, if the SPF implementation is defective
   and returns spurious "permerror" results, only the sender is actively
   notified of the defect (in the form of rejected mail), and not the
   receiver making use of SPF.

   The less intrusive handling choice is to deliver the message, perhaps
   with some kind of annotation of the difficulty encountered and/or
   logging of a similar nature. However, this will not be desirable to
   operators that wish to implement SPF checking as strictly as
   possible, nor is this sort of passive problem reporting typically
   effective.

   There is of course the option placing this choice in the hands of the
   operator rather than the implementer since this kind of choice is
   often a matter of local policy rather than a condition with a
   universal solution, but this adds one more piece of complexity to an
   already non-trivial environment.

   Both implementers and operators need to be cautious of all choices
   and outcomes when handling SPF results.

Appendix H. **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 slowly evolved until the spring of 2004 and has since
   stabilized. 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. The protocol was
   originally defined in [RFC4408], which this document replaces.

   [RFC4408] was designed to clearly document the protocol defined by
   earlier draft specifications of SPF as used in existing
   implementations. This updated specification is intended to clarify
   identified ambiguities in [RFC4408], resolve techincal issues
   identified in post-RFC 4408 deployment experience, and document widely
   deployed extensions to SPF that have been developed since [RFC4408]
   was published.

   This document updates and replaces RFC 4408 that was part of a group
   of simultaneously published Experimental RFCs (RFC 4405, RFC 4406,
   RFC 4407, and RFC 4408) in 2006. At that time the IESG requested the
   community observe the success or failure of the two approaches
   documented in these RFCs during the two years following publication,
   in order that a community consensus could be reached in the future.

   SPF is widely deployed by large and small email providers alike.
   There are multiple, interoperable implementations.

   For SPF (as documented in RFC 4408) a careful effort was made to
   collect and document lessons learned and errata during the two year
   period. The errata list has been stable (no new submissions) and
   only minor protocol lessons learned were identified. Resolution of
   the IESG's experiment is documented in [RFC6686].

Appendix I. Change History

   Changes since RFC 4408 (to be removed prior to publication)

      Moved to standards track

      Authors updated

      IESG Note regarding experimental use replaced with discussion of
      results

      Process errata:

      Resolved Section 2.5.7 PermError on invalid domains after macro
      expansion errata in favor of documenting that different clients
      produce different results.

      Add %v macro to ABNF grammar

      Replace "uric" by "unreserved"

      Recommend an SMTP reply code for optional permerror rejections

      Correct syntax in Received-SPF examples

      Fix unknown-modifier clause is too greedy in ABNF

      Correct use of empty domain-spec on exp modifier

      Fix minor typo errata

      Convert to spfbis working group draft,
      draft-ietf-spfbis-4408bis-00

      Addressed Ticket #1, RFC 4408 Section 2.5.6 - Temporary errors by
      giving the option to turn repeated SERVFAIL into permerror and
      adding RFC 2308 reference.

      Clarified text about IPv4 mapped addresses to resolve test suite
      ambiguity

      Clarified ambiguity about result when more than 10 "mx" or "ptr"
      records are returned for lookup to specify permerror. This
      resolves one of the test suite ambiguities

      Made all references to result codes lower case per issue #7

Adjusted section 2.2 Requirement to check mail from per issue #15

Added missing "v" element in macro-letter in the collected ABNF per issue #16 - section 8.1 was already fixed in the pre-WG draft

Marked ptr and "p" macro SHOULD NOT use per issue #27

Expunged lower case may from the draft per issue #8

Expunged "x-" name as an obsolete concept

Updated obslete references: RFC2821 to RFC5321, RFC2822 to RFC5322, and RFC4234 to RFC5234

Refer to RFC6647 to describe greylisting instead of trying to describe it directly.

Updated informative references to the current versions.

Start to rework section 9 with some RFC5598 terms.

Added mention of RFC 6552 feedback reports in section 9.

Added draft-ietf-spfbis-experiment as an informational reference.

Drop Type SPF.

Try and clarify informational nature of RFC3696

Fix ABNF nits and add missing definitions per Bill's ABNF checker.

Make DNS lookup time limit SHOULD instead of MAY.

Reorganize and clarify processing limits. Move hard limits to new section 4.6.4, Evaluation Limits. Move advice to non-normative section 9.

Removed paragraph in section 10.1 about limiting total data volumes as it is unused (and removable per the charter) and serves no purpose (it isn't something that actually can be implemented in any reasonable way).

Added text and figures from Alessandro Vesely in section 9.1 to better explain DNS resource limits.

Multiple editorial fixes from Murray Kucherawy's review.

Also based on Murray's review, reworked SMTP identity definitions
and made RFC 5598 a normative reference instead of informative.
This is a downref that will have to be mentioned in the last call.

Added RFC 3834 as an informative reference about backscatter.

Added IDN requirements and normative reference to RFC 5890 to deal
with the question "like DKIM did it.:

Added informative reference to RFC 4632 for CIDR and use CIDR
prefix length instead of CIDR-length to match its terminology.

Simplified the exists description.

Added text on creating a Authentication-Results header field that
matches the Received-SPF header field information and added a
normative reference to RFC 5451.

Added informative reference to RFC 2782 due to SRV mention.

Added informative reference to RFC 3464 due to DSN mention.

Added informative reference to RFC 5617 for its DNS wildcard use.

Clarified the intended match/no-match method for exists.

Added new sections on Receiver policy for SPF pass, fail, and
permerror.

Added new section 9 discussion on treatment of bounces and the
significance of HELO records.

Added request to IANA to update the SPF modifier registry.

Substantially reorganized the document for improved readability
for new users based on WG consensus.

Author's Address

    Scott Kitterman
    Kitterman Technical Services
    3611 Scheel Dr
    Ellicott City, MD 21042
    United States of America

    Email: scott@kitterman.com