

Caller ID for E-Mail

The Next Step to Deterring Spam

Microsoft Corporation

Published: February 12, 2004

Over recent years, the rate and frequency at which computer users receive unwanted and unsolicited e-mail (colloquially known as “spam”) has increased significantly. In many regions the problem has become so severe so as to perhaps threaten the viability of e-mail as a useful communication medium. It is time that something be done. This paper is a draft-for-comment of an initial approach which can begin to address this issue. The approach is analogous to the caller id technology found in the telephone system. Although its details are still in flux and not finalized, the technical design has to date already received considerable review and scrutiny. Comments, questions, suggestions and other feedback regarding the proposal are actively solicited. We believe that through coordinated action by the participants in the Internet e-mail community, significant progress in deterring spam can be made, and computer users everywhere will experience significant reductions in undesired and unwanted e-mail correspondence.

Copyrights, Licenses, Etc.

Copyright © 2004 by Microsoft Corporation. All rights reserved

Microsoft Corporation (the “Author”) hereby grants you permission to copy, review, evaluate, publish and distribute this Caller ID for E-mail Specification (the “Specification”) as a reference to assist you in planning and designing your product, service, or technology. All other rights are retained by the Author. You may not: (i) modify any part of the Specification, (ii) remove this notice or any license terms related to this Specification, or (iii) give any part of this Specification, or assign or otherwise provide your rights under this notice, to anyone else.

The Author believes that it has patent rights (patent(s) and/or pending applications(s)) that relate to the technologies discussed in the Specification. The Author is prepared to grant a royalty-free patent license with other reasonable and non-discriminatory terms to individuals and organizations interested in implementing the Specification, as set forth in the Caller ID for E-mail Implementation License, available at http://www.microsoft.com/mscorp/twc/privacy/spam_callerID.mspx.

THE SPECIFICATION MAY CONTAIN PRELIMINARY INFORMATION OR INACCURACIES, AND IS PROVIDED “AS IS,” AND THE AUTHOR MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS, STATUTORY, OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE SPECIFICATION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH SPECIFICATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

THE AUTHOR WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OF THE SPECIFICATION EVEN IF THE AUTHOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES IN ADVANCE.

The name and trademarks of the Author may NOT be used in any manner, including advertising or publicity pertaining to the Specification or its contents without specific, written prior permission. Title to copyright in the Specification will at all times remain with the Author.

No other rights are granted by implication, estoppel or otherwise.

Feedback

The Author of the Specification welcomes public feedback and review and believes that such feedback can strengthen and improve the Specification. If you want to provide comments, questions, suggestions and other such feedback on the Specification (“Feedback”), you agree to make such Feedback available in accordance with the terms below. You should direct such Feedback to LessSpam@Microsoft.com.

You have no obligation to give the Author any Feedback. However, any Feedback you voluntarily provide may be used by the Author in any product, service, or technology (collectively, “Author Offerings”) which in turn may be relied upon by other third parties to develop their own products. Accordingly, if you do give Author any Feedback, you agree: (a) Author may freely use, reproduce, license, distribute, and otherwise use in any Author Offerings; (b) you also grant third parties, without charge, only those patent rights necessary to enable other products that use or interface with any specific parts of an Author Offering that incorporates your Feedback; and (c) agree to not give Author any Feedback (i) that you have reason to believe is subject to any patent, copyright, or other intellectual property claim or right of any third party; or (ii) subject to license terms which seek to require any Author Offering incorporating or derived from such Feedback, to be licensed to or otherwise shared with any third party.

Table of Contents

1. Caller ID for E-mail	3
2. Technical Preliminaries	5
2.1. Terminology	5
2.2. Storing E-mail Policy Information in DNS.....	5
3. Technical Details: Preventing Forged Domains in E-mail	7
3.1. Publication of Outgoing Mail Servers	7
3.1.1. Examples of Outbound Mail Servers Published In E-mail Policy Documents	10
3.1.2. Bringing Outbound Mail Servers Online and Offline	11
3.2. Mapping a Message to a Purported Responsible Domain.....	11
3.2.1. Implications for Mobile Users	12
3.2.2. Implications for Mailing Lists.....	13
3.2.3. Implications for Mail Forwarders	13
3.2.4. Implications Regarding Multiple Apparent Responsible Identities	14
3.2.5. Summary of Implications.....	14
3.3. Correlation and Checking of Purported Responsible Domains.....	15
3.3.1. Checking Purported Responsible Domains in E-mail Clients, Etc.	16
3.4. Domains Which Only Send Mail Directly to Recipients	18
3.5. Security Considerations of Caller ID for E-mail.....	19
3.5.1. DNS Security Considerations	19
3.5.2. SMTP Security Considerations.....	20
4. Related XML Schema	22
5. References	28

1. Caller ID for E-mail

As of mid-2003, about **83%** of the e-mail messages received by Microsoft® Hotmail® on a typical day are spam, unwanted and unsolicited e-mail sent indiscriminately to users. That's around **2.5 billion** out of nearly **3 billion** messages, and the numbers keep climbing. While spam has been around virtually as long as there has been e-mail, in the early years the scale involved was small enough so as not to be a significant practical annoyance or problem. That's all changed now. In recent times, the rate at which spam has been in users' electronic mailboxes has skyrocketed. The problem has become one of significant proportions, costing users, industry, and the economy at large billions of dollars annually in terms of lost human productivity and wasted computer resources. Left unchecked, spam threatens the viability of e-mail as a useful communication medium.

It is time to put a stop to this problem. Since the spammers aren't going to change their behavior, it falls on the rest of us to change what we ourselves do so that the majority of the spammers no longer have a viable and profitable business.

To control spam — to identify it and filter it — a variety of approaches have been developed. Although different on the surface, these approaches share some common characteristics. They all ask certain questions about an e-mail message. They all use the answers to help classify the message as legitimate or junk. One of the most important of these questions is simply “who is sending me the message?” With this information in hand, one can ask follow-up questions such as:

- Do we know this person?
- Do they have a history of sending legitimate e-mail or junk e-mail?
- Do we trust them?

It turns out that answering with certainty this one simple question “who is sending me the message?” is very problematic. When e-mail is transmitted over the Internet from one organization to another today, no authentication of the sender of the e-mail or the computers delivering it on the sender's behalf occurs. That is, no verification is done to ensure that mail which purports to be sent from, say, `someone@example.com` does in fact originate from computers under the control of the “Example” organization. It is a trivial matter for virtually anyone with a computer connected to the Internet to send mail and appear to be someone else in doing so. This is a form of forgery, and is often politely called “spoofing”. Automated infrastructure for preventing such forgeries is lacking: short of a human forensic investigation, there is no means to distinguish such mail from the real thing.

Needless to say, spammers routinely exploit this capability. Consequently, filtering spam is an error-prone activity. Junk e-mail that appears to originate from legitimate senders slips through filters. Worse, e-mail from legitimate senders is often blocked because their e-mail addresses have been spoofed and their reputations tarnished.

One way to begin to address this state of affairs is to mirror what the telephone system provides with caller id. In the telephone system, caller id technology reliably informs the receiver of a phone call as to which telephone number is on the other end of the line. This is often related to the human being you end up talking to, but is not one-and-the-same concept. A more direct analogy to e-mail is most evident if one considers the transmission of faxes over the telephone: caller id provides the phone-company-provided phone number of the party sending the faxes, but this is a separate notion from the identity of the party who is responsible for the contents of the fax itself. That said, in most cases one expects the two notions to be related, and would be suspicious if they were not.

In the world of e-mail on the Internet today, we're missing this sort of caller id mechanism. However, as has been observed by others (see references [1], [2] and [3]), a caller id mechanism can be achieved in e-mail relatively simply by having administrators of domains publish the Internet addresses of their *outgoing* e-mail servers in the Domain Name System (DNS) in addition to the *incoming* e-mail servers that they list there today (the term “domain” refers to the part of an e-mail address that follows the at sign “@”). DNS is a planet-wide, distributed database that provides numerical Internet Protocol (IP) addresses and other information about Internet domains, much as a paper telephone book lists addresses and telephone numbers under the names of telephone subscribers. When e-mail is transmitted from one organization to another, the computers of the sending organization need to determine which computer the receiving organization has designated as the one that handles its incoming mail. For example, the DNS entry for the domain `example.com` might include the fact that the computer at Internet address `111.222.123.121` is one to which mail destined for e-mail addresses `@example.com` may be delivered. To actually deliver mail to this organization, the sending computer opens up a connection to `111.222.123.121` and

hands over messages one-by-one using the conversational rules of a standard known as the Simple Mail Transfer Protocol, or SMTP.

With the information listing outgoing mail servers in place, software that receives an incoming message can now verify that the computer used to send the message was in fact under the control of the domain from which the message purports to have originated, just as (in analogy) the receiver of a fax sent over the phone can use caller id information to check that the phone number from which the fax was transmitted is consistent with what the contents of the fax actually say. With this infrastructure in place, e-mail software will be able to distinguish, for example, whether a message that claims to be from `someone@example.com` was actually sent from the `example.com` organization or not. If not, the mail is a forgery.

Note that the caller id mechanism defined here is not the equivalent of fully authenticated e-mail: there still remains, for example, the question as to whether the user named `someone` did in fact send the e-mail or whether, say, some rogue administrator at `example.com` sent the e-mail instead. Even so, having reliable domain information is a huge step forward, and will be of great utility to e-mail filtering and classification systems, allowing them to distinguish between spam and non-spam e-mail with greater certainty. This in turn allows organizations to more harshly penalize e-mail they believe to be spam with less fear of misclassification or “false positives”.

The caller id mechanism proposed here was designed with many considerations in mind. Among the more important of these were the following:

1. **Ease of adoption.** A key goal of this proposal is rapid and broad adoption. Therefore, it is a requirement that *wherever possible* any technical design operate within the capabilities, behaviors, and limitations of existing software. Clearly, it will not be possible to deploy this idea without some changes to some software. However, the proposal has been designed to keep such changes to a minimum.
2. **Scalability.** The design must scale up to meet the needs of the largest Internet service providers (ISPs) and down to the smallest office or home e-mail server. Specifically, it must support organizations that have hundreds or even thousands of e-mail servers as well as those that have just one. It must also support those who outsource their e-mail servers to another organization.
3. **Fairness.** The design must fairly distribute the costs of compliance. Today the costs of detecting and remedying the forging of addresses are placed entirely on the receiving organization; the design must rectify this imbalance. Organizations that want to ensure mail from their domains is not mistakenly judged to be spoofed should bear part of the cost burden. Organizations that want to accurately determine whether or not messages they receive have been spoofed should also bear a corresponding cost.
4. **Openness.** The technical design must be openly published so that any organization wanting to comply with its provisions may do so.
5. **Extensibility.** The technical design must allow for the publication of new or additional information about an organization’s e-mail policies and practices should this be required in future. It is important that this new information can be published without the need for the publication to be coordinated through a central body in order to, for example, avoid semantic collision with new information published by others.

Caller ID for e-mail does not prevent spam from being sent. But it does make it easier to detect, and provides a basis for further measures which can do an even better job at such detection. As these measures are implemented and incrementally deployed more and more broadly, it will become more and more difficult to make a lucrative living as a spammer. We will never make the problem go away entirely: any system powerful enough to do that will necessarily unduly impinge on the free exchange of legitimate communication, making the solution worse than the problem it was designed to correct. But what we can do is reduce the problem back to the manageable and tolerable level it once was at. It’s time we started.

2. Technical Preliminaries

Before diving into the technical details of the caller id mechanism, we begin with some necessary preliminaries.

2.1. Terminology

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

2.2. Storing E-mail Policy Information in DNS

At various points below and elsewhere, it is necessary that new kinds of policy and other information regarding the e-mail policies of a DNS domain be published. This section defines an approach to publishing such information.

The natural mechanism with which to publish information regarding a DNS domain is of course to use DNS itself in a straightforward, traditional manner. That said, unfortunate logistical complications are found in the selection of DNS resource record types that must be used to represent such information. A seemingly-obvious choice is to select appropriately-defined new record types for such new information. However, the use of new record types generally requires that DNS client software, server software, and administration tools all be updated to be made aware of the new types. Were we to require such updates here, the adoption of this proposal would be significantly impeded.

Accordingly, an alternate means of representing the new information has been chosen, one that only makes use of existing, commonly understood DNS resource record types. Specifically, XML-encoded information stored in the TXT resource records in the `'_ep'` subdomain of a DNS domain in question is used. These records form a record set, which can thus be retrieved with a single DNS query, an important performance concern.

The use and interpretation of this record set is as follows. Each TXT record in this record set **MUST** be of at most 2K characters in length (this accommodates restrictions of deployed DNS clients and servers, specifically broadly deployed versions of BIND). If there is just a single record in the record set, then the E-mail Policy Document for the domain is simply the sequence of strings in that one record concatenated together. Otherwise, the concatenated sequence of strings in each TXT record in the record set **MUST** begin with two distinguished characters (the two-digit decimal representation (with leading zeros as needed) of a non-negative integer is recommended) which is unique within the record set. Upon retrieval, the TXT records in the record set are sorted in ascending lexicographical order by these two characters, the first two characters removed from each record, and the results concatenated together to form one overall sequence of characters which is the E-mail Policy Document for the domain.

The E-mail Policy Document for the domain **MUST** be a valid UTF8-encoded XML document; in order to promote interoperability, other XML character encodings **MUST NOT** be used.

The XML E-mail Policy Document of a domain **SHOULD** contain information which the administrators of the domain attest is pertinent to the policies under which the domain receives or sends mail. The XML schema definition which **SHOULD** be used for such XML documents is found in §4; an E-mail Policy Document containing elements conforming to a different schema **SHOULD** be ignored by those applications which do not understand it.

XML E-mail Policy Documents **SHOULD** be cached in the normal manner for information returned from DNS. In addition, applications may find performance benefits from also caching at a higher semantic level the information derived from parsing these documents, but whether they in fact do so or not is their own local concern.

More will be said later regarding the details of E-mail Policy Documents, but the simple cases are quite simple: one writes the XML, it fits in a single TXT record, and this is copied to the DNS configuration file. For example (see also §3.1.1), the following illustrative fragment of a DNS configuration file shows an XML E-mail Policy Document indicating that the outbound mail servers of the domain `example.com` are exactly whatever the domain's inbound mail servers are. Indeed, for those domains where this is the applicable policy, this literal boilerplate TXT record can be used verbatim.

```
@ IN SOA ns.example.com. postmaster.example.com. (1 36000 600 86400 3600)
  _ep TXT ("<ep xmlns='http://ms.net/1' ><out><m><mx/></m></out></ep>")
```

This next slightly more contrived example illustrates how an E-mail Policy Document can be split across possibly several TXT records:

```
@ IN SOA ns.example.com. postmaster.example.com. (1 36000 600 86400 3600)
  _ep TXT ("02.3.4</a>"
           "      </m>"
           "      <m><mx/></m>"
           "    </out>"
           "</ep>" )
  TXT ("01<ep xmlns='http://ms.net/1' >"
       "  <out>"
       "    <m>"
       "      <a>1.2" )
```

Here, the e-mail policy information of the `example.com` domain is the following XML document (note that, per the usual XML rules, with the exception of the space character between `ep` and `xmlns`, whitespace is not significant):

```
<ep xmlns='http://ms.net/1' >
  <out>
    <m>
      <a>1.2.3.4</a>
    </m>
    <m><mx/></m>
  </out>
</ep>
```

3. Technical Details: Preventing Forged Domains in E-mail

As was mentioned previously, when e-mail is handed off today from one organization to another, as a rule absolutely no authentication of the sender of the e-mail or the computers delivering it on the sender's behalf takes place. Fortunately, there are some simple steps which can be taken to significantly alleviate this problem, steps which mimic within the e-mail infrastructure the caller id mechanism found in today's telephone system. Specifically, based on the ideas of Hadmut Danisch, Meng Weng Wong, and others (see references [1], [2], and [3]), the present proposal specifies that in addition to today's practice of publishing in DNS the addresses of their *incoming* mail servers, administrators of domains SHOULD also publish the addresses of their *outgoing* mail servers, the addresses of the computers from which mail legitimately originating from that domain will be sent. This information will then be used in enhancements to software that receives incoming mail to verify that computers handing off a message to them in fact are authorized to do so by the legitimate administrator of the domain from which the message is purported to have been sent.

From a technical perspective, the caller id mechanism is roughly separable into three parts:

1. the publication in DNS of the **outgoing mail servers** for a given domain in addition to the *incoming* mail servers which are already published today,
2. the identification of a particular **domain as being responsible** for a given message, and
3. the **correlation and checking of these** two pieces of information to determine whether the IP address from which the message was received was reasonable or not.

The following subsections examine each of these parts in turn. Significant subtleties and surprises lurk in each: readers are cautioned to read and consider this information carefully.

3.1. Publication of Outgoing Mail Servers

In this proposal, the identity of the outgoing mail servers for a domain SHOULD be published in the XML E-mail Policy Document for the domain, specifically in the `ep/out` element. If the XML E-mail Policy Document does not exist for a domain or the document exists but does not contain either an `ep/out/m` element or an `ep/out/noMailServers` element, then the domain makes no statement about its outbound mail servers. If the document exists and contains an `ep/out/noMailServers` element, then the domain is making the statement that there are *no* outbound servers for that domain.

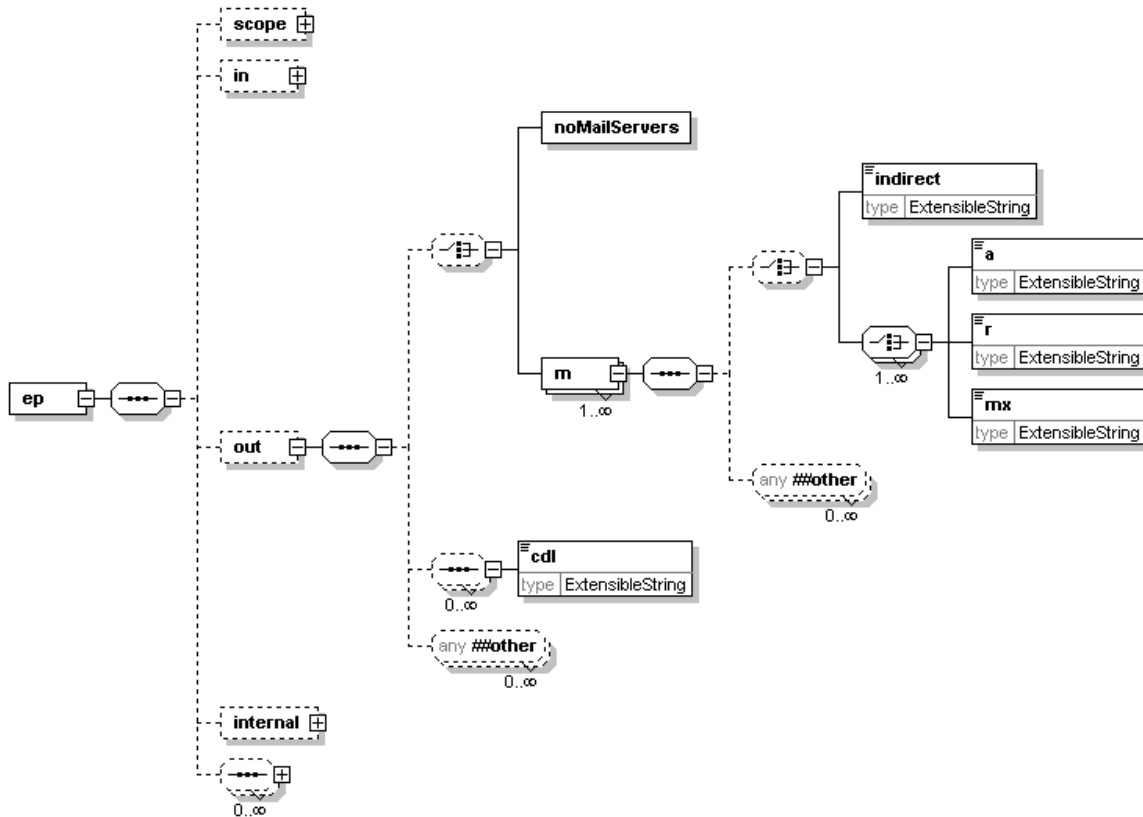
The portion of the E-mail Policy Document relevant to outbound mail servers can be depicted as shown in the diagram below (a more complete exposition of this schema is found in §4). Note that XML attributes (notably those on the element `m`) are not illustrated in this diagram; only XML elements are shown.

The outbound mail servers for the domain are listed as one or more occurrences of an `m` element within the element `ep/out`. The interpretation of each `m` element ultimately provides two essential pieces of information about the server in question:

1. whether or not it makes use of the "enhanced SMTP noop" functionality (described in §3.5.2 below) when making outbound SMTP connections, and
2. the IP address or addresses of the server.

Within the design, several important points were considered, including:

- a. Some domains (many of them well known) have literally thousands of outgoing mail servers. Thus, a means must be provided by which the need to list the address each server individually can be avoided. This is addressed by permitting legal *address range* (using address prefix list information in the style of in RFC3123) to be listed in place of individual addresses.
- b. Although not as common as it once was, SMTP outbound relaying must be accommodated. In such situations, the outbound servers that actually perform the inter-domain delivery of mail (as seen by the receiving domain) may not be administratively under the direct control of the sending domain. A means of indirection is thus provided, wherein a sending domain can avoid having to repeatedly and continually update its XML E-mail Policy Document to reflect the now-current addresses of the servers used by the relaying organization.



From the point of view of specification, the set of IP addresses from which mail from a given domain d may be transmitted to a receiving domain is denoted as $OutGoing(d)$. The function $OutGoing(d)$ is defined as follows.

1. If there exists an element $ep/out/noMailServers$ in the XML E-mail Policy Document of d , then $OutGoing(d)$ is defined to be the empty set.
2. If there exists at least one element $ep/out/m$ in the XML E-mail Policy Document of d , then $OutGoing(d)$ is defined to be the union over all the $ep/out/m$ elements m in the XML E-mail Policy Document of d of the set of IP addresses $OutGoing(m,d)$ mapped to by each.
3. Otherwise, the result of the function $OutGoing(d)$ is undefined.

In turn the function $OutGoing(m,d)$ is defined as follows. If m is an element of the form $ep/out/m$ in domain d , then $OutGoing(m,d)$ is the set of IP addresses mapped to by m as calculated in the following manner.

1. If the child element $m/indirect$ exists, then it MUST contain a fully qualified domain name d' . $OutGoing(m,d)$ is then defined as follows:
 - a. If d' contains an E-mail Policy Document in its DNS entries, then $OutGoing(m,d)$ is defined to be equal to $OutGoing(d')$.
 - b. Otherwise, $OutGoing(m,d)$ is defined to be the same set of addresses that are associated with the inbound mail servers of the domain d' as determined by the usual MX and A record processing (see §5 of RFC2821).
2. Otherwise, if any child elements m/a , m/r , or m/mx exist, then $OutGoing(m,d)$ is defined to be the set difference $A_+(m,d) - A_-(m,d)$ of the two sets $A_+(m,d)$ and $A_-(m,d)$ which respectively represent explicitly listed addresses and explicitly listed address exclusions for m . These two sets are specified as follows:

The set $A_+(m,d)$ is defined to be the union over all child elements m/a , m/r , or m/mx of the contributions of each as defined in the following manner:

- a. An m/a element MUST contain one of the following
 - i. the (usual) dotted-decimal textual representation of an IPv4 address as defined by §3.4.1 of RFC1035 (for example, “11.22.33.44”); in this case, the contribution to the set $A_+(m,d)$ is the IPv4 so noted.
 - ii. any of the three forms of the textual representation of an IPv6 address as defined by §2.2 of RFC2373 (for example, “1080:0:0:0:8:800:200C:417A” or “::FFFF:129.144.52.38”); in this case, the contribution to the set $A_+(m,d)$ is the IPv6 address so noted.
 - iii. a fully qualified domain name d' . In this case, the contribution to the set $A_+(m,d)$ is the set of IPv4 and / or IPv6 addresses listed in DNS for the domain d' using (respectively) A and / or AAAA records.
 - iv. an empty string. Here, the contribution to the set $A_+(m,d)$ is the set of IPv4 and / or IPv6 addresses listed in DNS for the current domain d using (respectively) A and / or AAAA records.
- b. An m/r element MUST contain the textual representation of a single prefix-denoted range of IP addresses as defined by §5 of RFC3123. Examples include “1:192.168.32.0/21” and “!1:192.168.38.0/28”. If the representation begins with a ‘!’ character, then its contribution to the set $A_+(m,d)$ is the empty set; otherwise, its contribution is the set of IP addresses denoted by the textual representation.
- c. An m/mx element MUST contain one of the following
 - i. a fully qualified domain name d' . In this case, the contribution to the set $A_+(m,d)$ is the same set of addresses that are associated with the inbound mail servers of the domain d' as determined by the usual MX and A record processing (see §5 of RFC2821).
 - ii. an empty string. Here, the contribution to the set $A_+(m,d)$ is the same set of addresses that are associated with the inbound mail servers of the current domain d as determined by the usual MX and A record processing (see §5 of RFC2821).

The set $A_-(m,d)$ is defined to be the union over all child elements m/r of the addresses indicated by each in the following manner:

- a. As was just noted, an m/r element MUST contain the textual representation of a single prefix-denoted range of IP addresses as defined by §5 of RFC3123. If the representation begins with a ‘!’ character, then its contribution to the set $A_-(m,d)$ is the set of IP address denoted by the textual representation but with the ‘!’ removed; otherwise, its contribution is the empty set.
3. Otherwise, $OutGoing(m,d)$ is defined to be the same set of addresses that are associated with the inbound mail servers of domain d as determined as determined by the usual MX and A record processing (see §5 of RFC2821).

Note that, as usual, internal DNS processing will automatically follow CNAME resource record aliasing should such aliases be used. Be aware that the use of the indirection mechanism or the use of the MX and / or A record processing per §5 of RFC2821 and the like will result in more DNS queries during the resolution than would otherwise be the case.

Implementations of this functionality should take internal measures to detect cycles of recursion as the evaluation progresses. While detecting actual recursion loops is RECOMMENDED, a simple approach is merely to count recursion depth; if that approach is used, then to promote interoperability a depth of at least eight recursions SHOULD be accommodated. If such a recursion cycle is detected, the results of the overall evaluation SHOULD be considered undefined, resulting in an inability to discern the outbound servers of the domain in question: the querying system SHOULD act in the same manner as if the outbound information were not published at all. If a DNS query involved in the evaluation times out or otherwise fails, then a reasonable approach is similarly to treat the query as unable to discern the outbound servers of the domain. Moreover, in order to deter a denial of service attack against querying mail servers through the use of deep and broad trees of indirection, a querying system MAY itself impose a time limit on the duration of the execution of the overall query including all its indirections. If present, such a limit is implementation-dependent, but SHOULD NOT be less than twenty seconds.

3.1.1. Examples of Outbound Mail Servers Published In E-mail Policy Documents

Here are some simple illustrative examples:

- (1) My outbound mail servers are whatever my inbound mail servers are:

```
<ep xmlns='http://ms.net/1'><out><m><mx/></m></out></ep>
```

- (2) My outbound mail server is at the single address 192.168.210.101:

```
<ep xmlns='http://ms.net/1'><out><m><a> 192.168.210.101</a></m></out></ep>
```

- (3) My domain has three particular outbound mail servers:

```
<ep xmlns='http://ms.net/1'><out><m>
  <a>192.168.210.101</a>
  <a>192.168.210.102</a>
  <a>192.168.210.107</a>
</m></out></ep>
```

- (4) My domain has no outbound mail servers:

```
<ep xmlns='http://ms.net/1'><out><noMailServers/></out></ep>
```

- (5) The outbound mail servers of my domain all live within one particular group of 16 IP addresses:

```
<ep xmlns='http://ms.net/1'><out><m>
  <r>192.168.210.101/28</r>
</m></out></ep>
```

- (6) My domain employs contoso.com to send mail on its behalf. In addition, my domain also sends mail outbound from its inbound servers and from the specific address 192.168.210.101:

```
<ep xmlns='http://ms.net/1'><out>
  <m><indirect>contoso.com</indirect></m>
  <m><mx/></m>
  <m><a>192.168.210.101</a></m>
</out></ep>
```

- (7) My domain has multiple subdomains, all of which use the same outbound mail servers.

- o Create CNAME records for each subdomain:

```
_ep.sub1.example.com  IN CNAME  outbound.example.com
_ep.sub2.example.com  IN CNAME  outbound.example.com
_ep.sub3.example.com  IN CNAME  outbound.example.com
```

- o In `_ep.outbound.example.com` (a dummy domain), register the required E-mail Policy Document.

- (8) My domain has hundreds of scattered outbound e-mail servers and will not open port 53 to TCP traffic. This can be accommodated by using `<indirect>` to dummy subdomains in order to build a tree of internal nodes, all of which can fit in a DNS UDP response, then populating the leaves of that tree with pieces of the necessary information. For example:

- o In `_ep.example.com` publish this E-mail Policy Document:

```
<ep xmlns='http://ms.net/1'><out><m>
  <indirect>list1.outbound.example.com</indirect>
  <indirect>list2.outbound.example.com</indirect>
</m></out></ep>
```

- o In `_ep.list1.outbound.example.com` (a dummy domain) publish a subset of my outbound server addresses information. For example:

```
<ep xmlns='http://ms.net/1'><out><m>
  <a>192.168.210.101</a>
  <a>192.168.210.102</a>
  <a>192.168.210.107</a>
  <a>    ...    </a>
</m></out></ep>
```

- o In `_ep.list2.outbound.example.com` (a dummy domain) publish the remaining outbound server address information. For example:

```
<ep xmlns='http://ms.net/1'><out><m>
  <a>192.168.210.253</a>
  <a>192.168.93.17</a>
  <a>192.168.93.21</a>
  <a>    ...    </a>
</m></out></ep>
```

- (9) The outbound servers of my domain reside at the address(es) of the domain `dynamic.example.com` as listed in DNS for that domain:

```
<ep xmlns='http://ms.net/1'><out><m><a>dynamic.example.com</a></m></out></ep>
```

3.1.2. Bringing Outbound Mail Servers Online and Offline

Domain administrators managing the publication of the identity of outbound mail servers for their domain should be cognizant of the fact that the time-to-live (TTL) value they publish on their records will adversely affect the speed with which a new outgoing mail server for the domain can be pragmatically brought online. Such effects can often be mitigated by the pre-publication of the IP addresses that such new servers will use, if such addresses can be predicted ahead of time. In particular, if the domain owns a particular range of IP addresses, and can administratively arrange that its new outgoing mail servers will use addresses in that range, the adverse effects can be eliminated by the one-time pre-publication of the address range.

An analogous issue arises involving the *retirement* of the addresses of *old* servers that are taken offline. Generally speaking, even after the last message has been transmitted from a retiring server address, there may be a period of time before all the messages sent from that address have been evaluated against a caller id check. This period of time may be considerable: if the caller id check is carried out in an e-mail client, the message may perhaps sit in a POP3 server for weeks or more while a user is on vacation. In order to bound this otherwise indeterminate length of time, it is required that a caller id check on a given message **MUST** be carried out within 28 days (672 hours) of the message having been received by the checking organization (that is, within 28 days of the message having crossed the inter-organizational SMTP hop); after such a period has elapsed, a caller id check **MUST NOT** be carried out. Organizations wishing to retire addresses of old mail servers **SHOULD** keep those addresses listed in their domain's E-mail Policy Document until this amount of time has elapsed since the last message sent from that address has crossed the inter-organizational SMTP hop and so been received by the destination domain. Once this time period has elapsed, the address can safely be removed from the E-mail Policy Document.

3.2. Mapping a Message to a Purported Responsible Domain

For each message transferred through SMTP from one organization to another (information regarding published outbound mail servers need not and likely ought not to be interrogated as mail is relayed within an organization, where presumably the trust relationships amongst the computers involved are such that they need not guard themselves against spam originating from within the organization itself) a ***purported responsible address*** is determined as the first from the following list of items which is both present and non-empty:

1. the first `Resent-Sender` header in the message, unless (per the rules of RFC2822) it is preceded by a `Resent-From` header and one or more `Received` or `Return-Path` headers occur after said `Resent-From` header and before the `Resent-Sender` header (see §3.6.6. of RFC2822 for further information on Resent headers),
2. the first mailbox in the first `Resent-From` header in the message,
3. the `Sender` header in the message, and
4. the first mailbox in the `From` header in the message.

Every well-formed e-mail message will contain at least one of the above headers. Any message that is sufficiently ill-formed as to not contain any of these **SHOULD** be considered very heavily suspect as spam (otherwise, spammers will just resort to omitting all of these headers). The particular listed order in which these headers are searched determines in a manner consistent with the defined semantics and usage of each a purported responsible address which is most immediately responsible for the transmission of a message. Roughly speaking (but see RFC2822 for details), these semantics can be characterized as follows:

Resent-Sender: addr1@example.com

The message was previously delivered to the mailbox of its final destination, but it was subsequently reintroduced into the e-mail transport system, and addr1@example.com was the agent that actually carried out the reintroduction on behalf of the party listed in the Resent-From: header (which per RFC2822 MUST be present) whose desire it was that this occur.

Resent-From: addr2@example.com

The message was previously delivered to the mailbox of its final destination, but it was subsequently reintroduced into the e-mail transport system, and addr2@example.com was the party whose desire it was to have the message reintroduced; that is, addr2@example.com is responsible for the message's reintroduction.

Sender: addr3@example.com

addr3@example.com is the agent responsible for the actual transmission of the message. That is, it was addr3@example.com that actually caused the message to be injected into the e-mail transmission system. For example, if a secretary were to send a message for another person, the mailbox of the secretary would appear in the Sender: field and the mailbox of the actual author would appear in the From: field.

From: addr4@example.com

addr4@example.com is the author of the message. That is, addr4@example.com is responsible for the writing of the message.

The *purported responsible domain* of a message is defined to be the domain part of the message's purported responsible address.

3.2.1. Implications for Mobile Users

Among the today's broad community of e-mail users are some who legitimately send e-mail from IP addresses which are not administratively connected with their home domain.

A common case is that of a mobile user, who may be using a laptop PC, an e-mail-enabled mobile phone, or other device. For example, suppose that Adam from example.com uses a portable e-mail messaging device to send much of his mail. Adam has an e-mail account adam@example.com that he wishes his recipients to think of as his 'actual' address. However, he also has an account adam@consolidatedmessenger.com with the system run by his mobile device provider, and, indeed, when he sends mail from his device the message travels directly to its destination using servers under the control of consolidatedmessenger.com; the servers of example.com are not involved. To make this situation function without appearing to have a spoofed domain, Adam should ideally (although it may not be technically possible today) configure his mobile e-mail messaging device so that the following headers appear in the message:

From: adam@example.com
Sender: adam@consolidatedmessenger.com

The purported responsible domain of Adam's mail is then consolidatedmessenger.com.

Many other mobile scenarios can be accommodated with an analogous approach, including that of Web-originated e-mail from greeting card sites and the like. That said, an alternate, different methodology is also applicable in many mobile scenarios. Since there is little legitimate point in having an e-mail address that can send mail but cannot receive it, there necessarily already exist servers which are administratively associated with the domains of each legitimate user, namely the servers which manage the receipt of the user's inbound mail. In many cases it is reasonable for mobile users to connect to these servers (or other administratively-related servers) to also relay their outbound messages. Adam might, for example, use his mobile e-mail messaging device in its alternate transmission mode wherein it communicates with his adam@example.com mailbox, which in turn sends the mail to its ultimate recipient in a manner indistinguishable from Adam having been sitting in front of his example.com mail reader in his home office. Where feasible, this approach is straightforward and is often to be preferred.

It is worth nothing in passing that it remains solely a decision for each individual domain such as example.com to choose whether or when to protect their reputation by taking the steps of §3.1. As more and more domains take such steps, spammers will necessarily migrate their spoofing to the remaining domains, which in turn may appear more and more suspect, but in the end each domain owner is empowered to act on their own behalf.

3.2.2. Implications for Mailing Lists

The core functionality provided by mailing list software is the provision of e-mail addresses (the address of each list) that take the mail received thereat and retransmit it on to the registered members of the list. In order to avoid the appearance of domain spoofing, such software SHOULD add a `Resent-From:` header to each retransmitted message to indicate that this action has been carried out. Alternately, a `Sender:` header MAY be used instead if one is not already present: indeed a number of existing mailing list agents already add such a `Sender:` header, and so need no adjustment to conform to this proposal.

Continuing our example, if Adam sends mail from his mobile e-mail messaging device to the mailing list `asrg@ietf.org`, when the message is transmitted to the list members it might look in part like the following

```
Resent-From: asrg@ietf.org
Resent-Date: Tue, 16 Dec 2003 14:33:13 -0800
Received: from ... by ietf-mx.ietf.org ...
...
To: asrg@ietf.org
From: adam@example.com
Sender: adam@consolidatedmessenger.com
```

Because of the `Resent-From:` header, the purported responsible domain of this message is `ietf.org`.

Note that the `Resent-Date:` header is included here per the requirements of RFC2822. Also note that the `Resent-*` headers are prepended *earlier* in the message than the corresponding `Received:` header; this is necessary in order to reliably demarcate the possibly multiple groups of `Resent-*` headers that may accrete as the message makes its way along its journey.

3.2.3. Implications for Mail Forwarders

Today many mail forwarding services exist, wherein mail sent to one e-mail address will be automatically forwarded to a second address. The details of exactly how such services operate differ from service to service, but a widespread practice is to carry out the forwarding by retransmitting messages *verbatim*, preserving exactly both original the SMTP-level envelope information as well as the entire original message body. That is, in the existing verbatim retransmission practice legitimate mail sent from any domain can be legitimately delivered to its recipient's systems from virtually any IP address, all in a manner entirely indistinguishable from the actions of a forger. If this behavior were to be preserved, any attempt to distinguish forgeries from legitimate mail would be problematic (see also the discussions of this issue in §10 of reference [1] and §3.4 of reference [2]).

Therefore, the practice of forwarding mail using verbatim retransmission is to be discouraged. We propose a very small modification to forwarding practices in order to allow forwarded mail to be distinguished from spam, namely that forwarding retransmission services SHOULD annotate the message as they carry out the retransmission action.

As before, this can be accomplished concretely by the inclusion by the forwarding agent of a new `Resent-From:` header in the message. Continuing our previous example yet again, suppose I subscribe to the ASRG mailing list using the address `bob@forwarderexample.com`, and further suppose that I have configured `bob@forwarderexample.com` to forward mail that it receives to `bob@example.com`. By the time Adam's message reaches me at this latter address where I actually read the mail, it's headers in part should look like the following.

```
Received: from ... by example.com ...
...
Resent-From: bob@forwarderexample.com
Received: from ... by forwarderexample.com ...
...
Resent-From: asrg@ietf.org
Resent-Date: Tue, 16 Dec 2003 14:33:13 -0800
Received: from ... by ietf-mx.ietf.org ...
...
To: asrg@ietf.org
From: adam@example.com
Sender: adam@consolidatedmessenger.com
```

The purported responsible domain of the message is now `forwarderexample.com`.

That all said, the legacy practice of verbatim retransmission can be accommodated to a limited degree at the expense of introducing narrow but entirely open holes in the forgery detection infrastructure. Fortunately, it is the case that most e-mail users are aware of the forwarding addresses that indirect to them, as they usually were involved in their establishment. In the above example, for instance, bob@example.com is likely aware of the existence of bob@forwarderexample.com's forwarding behavior, since Bob probably set up the forwarding relationship in the first place. Given this state of affairs, it is not unreasonable that the user of the forwarded-to system be provided with mechanisms in his e-mail software by which the filtering actions which would otherwise be carried out for the forwarded mail may be overridden. Specifically, the original destination address (here bob@forwarderexample.com) might be listed as a "known recipient" for bob@example.com: if this address appears in the To: or Cc: lines of mail delivered to bob@example.com, then the message would not be subjected to normal forgery detection processing. The forwarded mail thus will get through; however, this can be exploited by spammers should they become aware of this relationship between these two addresses. Such controlled and focused disabling of the forgery analysis strikes a balance between accommodating legitimate legacy forwarding systems on the one hand and defeating domain-spoofing spammers on the other.

3.2.4. Implications Regarding Multiple Apparent Responsible Identities

Each of the four categories of identification of §3.2 used in determining the purported responsible address for a message has slightly different semantics, the details of which are articulated in RFC2822 (see "§3.6.2 Originator Fields" and "§3.6.6 Resent Fields"). It is quite possible that a given piece of received mail contains a different address or addresses within the headers of each of these different four categories. As has been seen, sometimes such a combination of identities is reasonable, while other times it is not.

The mechanisms and policies of §3.1 through §3.2.1 determine the purported party who was most immediately responsible for the transmission of a message and verify that the message has not been spoofed with respect to the domain of that party. Moving beyond this to consider the general relationship among the various possible categories of purported responsible address is generally beyond the scope of this proposal per se, though §3.4 does address one important case. More properly such is a role of mail filtering and e-mail client software within a receiving system. That said, we do here offer some suggestions regarding how that might work.

Common historical practice in mail reading software regarding the mail originator and resent headers has been to present only the contents of the From: header to the users; the other related headers (Sender:, Resent-From:, Resent-Sender:) have not been shown. This behavior SHOULD change. Messages with combinations of identities in the originator headers SHOULD be rendered differently than messages in which the identities are the same. Specifically, it is RECOMMENDED that if the purported responsible addresses of a message is not the same as the address that would be rendered as the From: address that both these addresses be exhibited to the user. For example, the message in the example from §3.2.3 might be presented by e-mail client software as being

From bob@forwarderexample.com on behalf of adam@example.com

or

From adam@example.com via bob@forwarderexample.com

instead of the historical

From adam@example.com

Indeed, this suggestion is essentially a straightforward generalization of the behavior that exists today in some newer mass market e-mail clients with respect to the processing of From: and Sender: headers.

3.2.5. Summary of Implications

Examples of the implications of the various scenarios described in the immediately previous sections, as well as some additional ones, can be summarized in the following table.

Scenario	Action Required	Action Required By	Example Headers
Ordinary e-mail	Publish outbound IP addresses	Sender	To: bob@cohowinery.com From: adam@example.com
	Perform caller id check	Receiver	
Mobile user (User:	Add Sender :	Sender's MUA or	From: adam@example.com

sends mail over a 3rd party carrier network (e.g. consolidatedmessenger.com) but wishes mail to appear to be from his/her regular corporate account)	header	carrier network's MTA	Sender: adam@consolidatedmessenger.com To: bob@cohowinery.com
Mailing lists	Add Sender : or Resent-From: header	MTA associated with the mailing list	From: adam@example.com Sender: asrg@ietf.org To: asrg@ietf.org - or - Resent-From: asrg@ietf.org Received: from ... by Sender: adam@consolidatedmessenger.com From: adam@example.com To: asrg@ietf.org
Mail forwarding	Add Resent-From: header	Forwarding MTA	Received: ... by cohowinery.com ... Resent-From: bob@forwardexample.com Received: ... by forwardexample.com ... From: adam@example.com To: bob@forwarderexample.com
Web-generated e-mail (E-vites, E-greets, mail this news article to a friend, etc.)	Add Sender : header	Web application server	From: adam@example.com Sender: articles@dailynews.com To: bob@cohowinery.com
Anonymous e-mail	Add Resent-From: header	"Anonymizing" MTA	Received: ... by anonexample.com ... Resent-From: admin@anonexample.com <intermediate Received: headers removed> ... From: randomname@anonexample.com To: bob@cohowinery.com
Outsourced e-mail service provider	Use <indirect> element of E-mail Policy Document to point to outsourced outbound IP addressees	Sender	From: sales@example.com To: bob@cohowinery.com Return-Path: admin@outsourcexample.com
"Bring your own IP." (User sends mail using their ISP's SMTP network but has their primary e-mail account elsewhere.)	Add Sender header	Sender's MUA, or ISP's MTA	From: adam@example.com Sender: adam@ispexample.com To: bob@cohowinery.com Return-Path: adam@example.com

3.3. Correlation and Checking of Purported Responsible Domains

Once a purported responsible domain p for a message m has been determined, the IP address from which m was received should be compared with the IP addresses from which mail transmitted by the domain p are expected.

To that end, the XML E-mail Policy Document of p is consulted with the evaluation of $OutGoing(p)$ to learn the set of possible IP addresses from which mail sent by this domain can be validly transmitted (note that an actual DNS lookup need not occur for each message: significant and even pro-active caching of this information can be carried out). This set of IP addresses is compared to the IP address of the SMTP client transmitting the message (as indicated by the actual TCP connection) in order to determine whether domain spoofing is occurring or not. The outcome of this determination then forms input to the receiving system's mail filter.

Moreover, if it is determined that spoofing is occurring then the receiving system SHOULD NOT (§6.1 of RFC2821 notwithstanding) send any delivery receipts or delivery status notifications with regard to this message. This policy avoids a form of denial-of-service attack wherein such error notification messages are maliciously sent to a third party.

3.3.1. Checking Purported Responsible Domains in E-mail Clients, Etc.

Complications arise when the verification of the legitimacy of a purported responsible domain is carried out in e-mail client software or other software not actually on the organizational edge. Specifically, it is in general quite difficult for such software to know the IP address from which mail was actually delivered into the organization (and against which, of course, the domain check is to be made). Software running on the organizational edge owns the TCP connection in question, and thus can easily find this information; however, this is not true for other software.

In many non-edge cases, though, this missing address information can be gleaned through careful and judicious processing of `Received:` headers (see §3.6.7 of RFC2822 and §4.4 of RFC2821). As mail is relayed from SMTP server to SMTP server, `Received:` headers noting each hop are prepended to the beginning of the message. Therefore, for messages received into a given organization, there is in general some initial prefix of the `Received:` headers which were added by the organization itself, with the remainder of these headers following this prefix having been already present in the message before it relayed into the organization's edge SMTP server. Because they were added by the organization itself, it is reasonable that software within the organization can trust the headers in this prefix set. In particular, software can reasonably trust the last header in this prefix (the one added by the organization's edge SMTP server), which quite often has contained within it exactly the TCP-reported IP address needed to perform the caller id check.

The thorny difficulty, of course, lies in partitioning the `Received:` headers into the trustable prefix and the untrustable suffix. In general, this is not reliably solvable: there will always continue to remain deployment situations where this information simply is not available, and thus the forgery check cannot be performed by software within the organizational interior. However, various approaches can be used to reduce these situations to a minimum.

3.3.1.1. Look for your inbound mail servers

A first such approach seeks to locate `Received:` headers which indicate that they were added by the *inbound mail servers* of the domain in question. While the formal syntax of a `Received:` header is quite liberal, generally speaking a `Received:` header does actually contain both the SMTP server sending and the SMTP server receiving the message. The former is listed as the 'From' address, while the latter is listed as the 'By' address. For example (but see §4.4 of RFC2821 for details):

```
Received: from sendingDomain.com ([1.2.3.4]) by mx.receivingDomain.com; Tue, 4
Nov 2003 14:03:14 -0800
```

An interesting observation is the following. Suppose that within your organization you find a `Received:` header in a message where the 'By' address is one of your inbound mail servers (or, more accurately, where the set IP addresses resolved to by the indicated 'By' address has non-empty overlap with the set published inbound mail addresses for your domain per §5 of RFC2821). Then it is reasonable to assume *for the purposes of domain spoof checking* that all headers up to and including the first such header (call it header r_{e0}) were in fact added by your organization. That this is reasonable *for this purpose* can be seen as follows:

1. If your domain does indeed add such a header r_{e0} , then the header r_{e0} and all previous headers were added by an organization you trust, and so the content of the headers can also be trusted.
2. If your domain does *not* add such a header r_{e0} , then if present in the message it must have been provided by an attacker from outside the organization in an attempt, presumably, to defeat the caller id check logic. Note, however, that even if the attacker instead did nothing at all (that is, didn't add r_{e0}) that the caller id check would not in this scenario have been performed, due to lack of knowledge of the IP address from which the mail was delivered into the organization. Thus, the attacker cannot achieve any benefit by adding r_{e0} that he would not have received anyway.

Once r_{e0} is located, then immediately subsequent headers may also be attributed as being added by your organization so long as at least one of the following is true:

- a. The header also has a 'By' address which is also one of your inbound mail servers
- b. The header has a 'By' address which resolves to an IP address set which contains only non-Internet routable addresses (see RFC1918).

Let r_e be the last header added by your organization as determined by this algorithm. Then, if the 'From' address of r_e contains a TCP-reported IP address (per the `TCP-info` production of §4.4 of RFC2821), it is reasonable to believe that that address is the address from which your organization received the message, and to perform the caller id check on the message using that address as input.

It is worth repeating that the algorithm of this first approach is not reliable, in that there are many ways in which an organization's systems may be configured so as to prevent software inside the organization which parses `Received:` headers from finding the desired IP address. In particular, there historically has been little technical incentive to ensure that one's inbound mail servers are evident from one's added 'By' addresses, and so it should not be surprising that some domains are not configured in this manner. Though it is usually easy to rectify this situation, the next section presents an alternative approach which may also be pursued.

3.3.1.2. Look for something your domain administrator tells you to

Locating the `Received:` header added by your organizational edge would be trivially easy if two conditions were met.

1. There existed some distinguished character string s that all edge SMTP servers in your organization placed in the `Received:` headers that they add to incoming messages.
2. The string s was *not* present in the `Received:` headers added by any of the non-edge SMTP servers in your organization

If these two conditions hold, then the `Received:` header added by your organizational edge is simply the first `Received:` header which contains the distinguished string s . Moreover, in practice, it is often quite straightforward to arrange that these conditions hold, often amounting to little more than using existing administrative infrastructure on the edge servers in question to add some newly concocted distinguished string. Having located the `Received:` header added by the organizational edge, the IP address from which the message in question was delivered into the organization can be located as described in the previous section.

A less straightforward issue is that of how to communicate to software in the interior of an organization that wishes to perform caller id checks (for example, e-mail client software, or internal e-mail servers) whether these conditions hold and, if so, what the distinguished string s is. From an operational perspective, this could be carried out in an ad-hoc internal non-standardized manner, since, after all, all the computers in question are under the control of a single administrative entity. That said, having a standardized mechanism for disseminating this information is valuable, as it enables and supports interoperation of software from different vendors.

To that end, the following mechanism is defined. The E-mail Policy Document may contain zero or more elements of the form `ep/internal/edgeHeader`, each of which contains an arbitrary string. If a non-zero number of such elements is present, then the set of strings contained in the collective set of such headers denotes the distinguished strings that are found in the `Received:` headers added by the edge SMTP servers of this domain. That is, within the domain in question, the first `Received:` header which contains any of these strings should be understood to be the header which was added by the edge.

For example the following E-mail Policy Document indicates that the edge SMTP servers of the domain (which is presumably `example.com`) always add a `Received:` header which contains either the string `>***example.com edge***` or the string `mx.example.com` and that the `Received:` headers by SMTP servers inside the organization never contain either of these strings.

```
<ep xmlns="http://ms.net/1">
  <internal>
    <edgeHeader>***example.com edge***</edgeHeader>
    <edgeHeader>mx.example.com</edgeHeader>
  </internal>
</ep>
```

3.3.1.3. Parsing Received: headers

Formally, the specification of the syntax of `Received:` headers of Internet mail messages is found in §4.4 of RFC2821. Unfortunately, actual practice exhibits considerable deviation from this specification, almost (but not quite) to the point of being free-form text. Naturally, that presents difficulty in parsing them to locate the ‘From’ and ‘By’ addresses therein. In practice, the following heuristics to locating this information have been found to be workable.

1. If the first case-insensitive whitespace-separated word of the header isn’t “From”, then the header cannot be successfully parsed.
2. Find the first occurrence of the case-insensitive whitespace-separated word “By” before the first semicolon, outside of any parentheses, brackets and quotation marks. If none exists, then the header cannot be successfully parsed.
3. In between, if anything looks like an IP address, that’s the ‘From’ address. An IP address, for these purposes, is four sets of digits, separated by periods, and optionally followed by a colon and another set of digits. Otherwise, if anything in between looks like a domain name, then *that’s* the ‘From’ address. A domain name, for these purposes, is sets of domain-legal characters separated by periods. It must have a least one period, and its last character must be alphabetic. Otherwise, the header cannot be successfully parsed.
4. If the first word after the “By” looks like a domain name, then that’s the ‘By’ address. Otherwise, the header cannot be successfully parsed.

3.4. Domains Which Only Send Mail Directly to Recipients

One important sub-case of the general issue discussed in §3.2.4 regarding multiple apparent responsible identities warrants special attention and consideration. As we explored previously, the life of an e-mail message as it flows from its original sender to its ultimate recipient can in general be a complex one, possibly involving many occurrences of the message being delivered to a mailing list or forwarding service where it is updated and subsequently reintroduced and sent on to a different destination.

However, not all mail is subject to the generality of such an arbitrary pattern of flow. In particular, there is an important volume of mail, largely that of business-to-consumer correspondence involving billing statements, account maintenance, and the like, which will never legitimately be sent to mailing-lists. From the sender’s perspective, such mail is being delivered directly to what they believe to be the ultimate recipient. If the organizations which have this policy can communicate that fact to mail receivers, then receiving systems can with greater confidence apply significantly pejorative penalties to mail they receive which is in violation of the policy.

Domains which have this kind of policy can indicate so in their E-mail Policy Document. Specifically, a domain owner can indicate that mail from their domain only sends mail directly to their ultimate recipients by including on the `ep/out` element of their E-mail Policy Document a `directOnly` attribute with a true value.

A receiving system SHOULD interrogate this information once they have determined for a given message that domain spoofing is not taking place. Specifically, if the Caller ID check passes, and the purported responsible domain of the message is different than the domain of the first mailbox in the `From` header in the message (the “From Domain”), then the receiving system SHOULD retrieve the E-mail Policy Document of the From Domain. If the document exists and contains a `directOnly` attribute on its `ep/out` element which has a true value, then the receiving system can conclude that the message is in violation of the delivery policy of its sender. As a consequence, the receiving system SHOULD apply significantly pejorative penalties to mail, unless it has additional local knowledge (such as known recipient lists) that might reasonably override that behavior.

Note that the policy of sending mail directly to recipients can only be expressed on the granularity of a domain. As a result, domain owners who wish to avail themselves of this facility will need to separate their direct-only mail from the rest and use a different domain for each. For example, if Humongous Insurance wished to use this mechanism, they might use `humongousinsurance.com` for the direct-only correspondence they have with their customers and something like `corp.humongousinsurance.com` as the domain by which their employees conducted their business with outside partners and collaborators. Alternately, `humongousinsurance.com` might be set up as the employee domain, and designated subdomains such as `sales.humongousinsurance.com` and `support.humongousinsurance.com` could be used for the direct-only customer correspondence.

3.5. Security Considerations of Caller ID for E-mail

There are some subtle consequences of the approach presented here for providing caller id information for e-mail. These we now explore (in no particular order).

3.5.1. DNS Security Considerations

The use of DNS described in §3.1 warrants careful examination from a security perspective.

The DNS protocol has a simple request-response design. DNS requests and responses are traditionally transported using UDP, though TCP is also broadly supported (see RFC1035). All DNS communications are carried out in a single message format; the same format is used in both requests and responses. The top level format of the message is divided into five sections, some of which are empty in certain cases:

1. a header,
2. a question for the DNS server,
3. some resources records answering the question,
4. some resource records pointing towards other servers that might be of better assistance in answering the question, and
5. resource records holding additional information.

An important aspect of the DNS architecture is that answers may be cached by clients so that subsequent requests to the server may be avoided; the duration over which this caching may be permitted is indicated by the server as part of each answer, and is known as a “time to live” (TTL).

One particular piece of information in the DNS header is used to match up responses to outstanding requests: a 16-bit identification field is provided by the requesting client, and this value is copied into the corresponding response by the DNS server (the questions in the request are similarly copied to the response). If a client is careful in the manner in which it reuses values for the identification field, this allows several DNS requests to be issued in parallel from a given *source IP address, source UDP port pair*.

The same mechanism, however, provides opportunity for a straightforward hijacking attack, though one difficult to achieve in practice: if an attacker manages to deliver a matching response for an outstanding request before the arrival of the legitimate response from the DNS server, such a fraudulent response will be indistinguishable from the legitimate one. The primary impediments to carrying out this attack include providing the correct value of the identification field, ensuring that the questions in the response match those in the request, and achieving the necessary packet timing.

If the DNS client software uses a good random number for the generation of the identification field, an attacker on average must attempt roughly 2^{16} responses differing significantly only in this field in order to be successful in achieving the correct value; all but one of these will be seen by the DNS client as ill-formed. Moreover, all these responses must be sent quickly, in order that they arrive before the legitimate response. These characteristics taken together provide a means of defense: DNS client software ought to monitor for such situations and consider themselves under attack if detected. Unfortunately, such sophistication is rare today.

Normally, it is virtually impossible for an attacker to successfully carry out a DNS hijack attack. In addition to the difficulty in matching the identification field value, attackers usually have no intrinsic means to know when a particular DNS client might happen to make a certain particular query request, so providing a corresponding fraudulent response before the DNS server does is very, very difficult.

However, if we are not careful, then depending on the particular usage scenario the same may not always hold true of the use of DNS as articulated above for publishing the identity of outbound mail servers. Specifically, if an attacker attempting to send domain-spoofed mail to an SMTP server also carries out a DNS hijacking attack on the server, he can to some degree use the timing of his delivery of the mail in order to help estimate when the receiving infrastructure will query DNS for the outbound mail server information of the domain in question. Moreover, given the specification of XML E-mail Policy Documents, the attacker knows to a very high degree the nature of the questions in these query requests. By using this timing and question-knowledge information, the attacker has an easier time succeeding than would otherwise be the case.

That said, important mitigating factors should be noted. First, while the architecture of DNS does indeed admit attacks, it is unclear whether it is economical for a spammer to exploit them, especially on a scale of any significant volume. Second, unlike many spamming activities, this particular form of attack is a criminal act in

many jurisdictions. Moreover, due to the nature of the DNS caching architecture, the execution of the attack tends to leave persistent evidence of its existence. Finally, the degree of vulnerability varies considerably depending on the particular usage scenario: for example, scenarios where the caller id check is performed in the organizational interior rather than on the organizational edge because there is a lesser timing correlation with the actions of the attacker. Fortunately, in these latter scenarios (the edge ones), there is already an expectation of SMTP queuing and delay, which allows us to take certain countermeasures.

With these considerations in mind, the following additional mitigating approaches may be employed:

1. Applications SHOULD where possible use DNS client software that monitors for and guards against attacks.
2. Especially for DNS queries for XML E-mail Policy Documents issued against DNS servers that reside outside of one's organization, applications SHOULD where possible specify to their DNS client software that TCP-based DNS queries be used and UDP-based queries be avoided (note that this approach is also driven by size considerations). If this technique is used, then attackers attempting to spoof the result of the query face the additional burden of guessing TCP/IP sequence numbers used within the connection. This significantly decreases the feasibility of a successful attack.
3. SMTP servers receiving mail SHOULD introduce a random delay between the determination that the outbound servers for a certain domain are required and the issuance of the actual DNS query that requests their addresses. Though clearly a greater variability in this delay provides greater protection, even variability on the order of several seconds is very worthwhile.
4. As XML E-mail Policy Documents cached by a DNS client on behalf of an SMTP server expire under the TTL caching policy indicated by the server, they MAY be proactively refreshed by the SMTP server on a schedule independent of the further reception of mail from the domains in question. For these domains, this decouples the timing of the XML E-mail Policy Document DNS queries from the timing of transmission of mail from those domains, thus making attacks more difficult.

The benefit to be achieved in adopting any one or all of these mitigating approaches needs to be balanced against the costs and impediments each has to successful and broad deployment (indeed, for these reasons, use of DNSSEC is not among the list of recommended mitigations).

We must also keep in perspective that the consequence of a successful attack is simply that some mail that ought to be considered to be spam may fail to be detected as such and thus appear with more prominence to a user; that is, such consequences are considerably less severe than attacks in other situations, such as computer viruses.

3.5.2. SMTP Security Considerations

In using the IP address as reported by TCP as part of the process of eliminating domain spoofing described above, there is a small concern that TCP hijacking may successfully be able to forge this address. While such hijacking is not easy and is somewhat expensive to achieve, particularly with reasonable TCP/IP implementations which well-randomize their initial TCP sequence numbers and monitor for sequence number guessing attacks, it is unfortunately the case that the SMTP protocol is of the nature that one can accomplish significant mayhem (specifically, one can send mail) by hijacking just the *sending* side of the two-way TCP connection; this is in contrast to many other protocols where hijacking of both directions of the connection is effectively necessary in order to wreak havoc, a much more difficult attack to carry out.

In order to alleviate the concern of such an attack, we propose here a small enhancement to the SMTP protocol wherein the SMTP client demonstrates to the SMTP server that it has actually received some unique data returned previously by the server. With such enhancements in use, one-sided TCP hijackings are no longer effective against SMTP.

Because this enhancement operates at the SMTP level, it can be used without the need to *rely* on particular safety-conscious behavior of the perhaps varying underlying TCP implementations on which the SMTP software may be deployed, thus providing an increase in overall resilience of the system to attack.

Specifically, prior to sending any SMTP MAIL commands, an SMTP client SHOULD send to its server an SMTP NOOP command (see §4.1.1.9 of RFC2821) with a parameter containing the following:

- the 40 case-insensitive characters of the hexadecimal encoding of the SHA1 hash of the entire data provided previously by the server in its EHLO response (that is, the entire sequence of characters matching the `ehlo-ok-rsp` production found in §4.1.1.1 of RFC2821).

By including a random string as part of its EHLO response, an SMTP server can have reasonable confidence that, upon the receipt of such a NOOP command, the SMTP client is in fact receiving its responses, and one-side TCP/IP hijacking is not occurring. Note that, as a security consideration, the SMTP server provides no indication in response to the NOOP as to whether a correct hash value was provided.

Use or support of this enhanced NOOP command on the part of SMTP clients (and servers) is wholly OPTIONAL: absent the need to send this NOOP, no e-mail or DNS server software need be updated on the client side in order to for an SMTP client to participate in the design proposed here for publication of outbound mail server information; this simplicity is a significant factor that ought to foster its adoption. In order to preserve this characteristic while also allowing for enhanced NOOP support where possible, we provide a means by which the administrator of a domain can indicate whether any or all of its outbound mail servers support the enhanced NOOP. Such administrative indication on the part of the legitimate domain owner defeats an attempt by a potential hijacker to merely pretend that the domain he wishes to impersonate merely does not have enhanced-NOOP-aware software.

The administrative mechanism makes use of the optional `eNoop` XML attribute which may be found on `ep/out/m` elements in XML E-mail Policy Documents (see §4 below for details of this XML schema). This attribute, if present, indicates published knowledge of the behavior of particular outbound mail server(s) with respect to their using the enhanced NOOP; if the attribute is absent, then no information is provided as to whether the outbound servers will use the NOOP or not.

The `eNoop` XML attribute is interpreted as follows. Let T be the set of four enumerated values $\{useUnknown, uses, doesntUse\}$. Let $Meld: T \times T \rightarrow T$ be the symmetric binary function defined by the following table:

	<i>useUnknown</i>	<i>uses</i>	<i>doesntUse</i>
<i>useUnknown</i>	<i>useUnknown</i>	<i>uses</i>	<i>doesntUse</i>
<i>uses</i>	<i>uses</i>	<i>uses</i>	<i>uses</i>
<i>doesntUse</i>	<i>doesntUse</i>	<i>uses</i>	<i>doesntUse</i>

Let D be the set of all fully qualified domain names, IP be the set of all IP addresses, and A be the set of all XML attributes of Boolean type. Define $Uses: D \times IP \times A \times T \rightarrow T$ to be the function where $Uses(d, ip, a, t)$ is computed as follows.

Let M_d be the set of all the `ep/out/m` XML elements m in the XML E-mail Policy Document of d such that ip is in the set $OutGoing(m, d)$. Then,

1. Let t_d be a variable initialized with the value t .
2. For each element m in M_d ,
 - a. If the XML child element `m/indirect` exists, then let $t_{indirect}$ be $Uses(d', ip, t)$ where d' is the fully qualified domain name contained therein; otherwise, let $t_{indirect}$ be *useUnknown*.
 - b. If the XML attribute a exists as a child of m , then let t_m be *uses* if the attribute is true and *doesntUse* if the attribute is false; otherwise, let t_m be *useUnknown*.
 - c. Update t_d to be the value $Meld(t_m, Meld(t_{indirect}, t_d))$.
3. $Uses(d, ip, t)$ is then defined to be the resulting value of t_d .

Given these definitions, whether or not the SMTP client at IP address ip which according to §3.1 is a legitimate outgoing mail server for domain d will use the enhanced NOOP functionality when connecting to SMTP servers is given by $Uses(d, ip, eNoop, useUnknown)$: if that value is *uses*, then an SMTP client purportedly from domain d which omits the use of enhanced NOOP should be considered suspect.

4. Related XML Schema

As was described above, the element `ep` from the following XML schema should be used to indicate e-mail policies associated with a given DNS domain or e-mail address. Example instances of this element follows the schema specification itself.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://ms.net/1" xmlns="http://ms.net/1" xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified" blockDefault="#all">
  <xs:complexType name="ExtensibleString">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="ep">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="scope" minOccurs="0">
          <xs:complexType>
            <xs:choice>
              <xs:element name="domain" type="ExtensibleString"/>
              <xs:element name="element" type="ExtensibleString"/>
              <xs:sequence>
                <xs:element name="message-id" type="ExtensibleString"/>
                <xs:element name="date" type="ExtensibleString" minOccurs="0"/>
              </xs:sequence>
              <xs:any namespace="##other" processContents="lax"/>
            </xs:choice>
            <xs:anyAttribute namespace="##other" processContents="lax"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="in" minOccurs="0">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="hashedPuzzle">
                <xs:complexType>
                  <xs:attribute name="nMin" type="xs:nonNegativeInteger"/>
                  <xs:anyAttribute namespace="##other" processContents="lax"/>
                </xs:complexType>
              </xs:element>
              <xs:any namespace="##other" processContents="lax"/>
            </xs:choice>
            <xs:anyAttribute namespace="##other"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

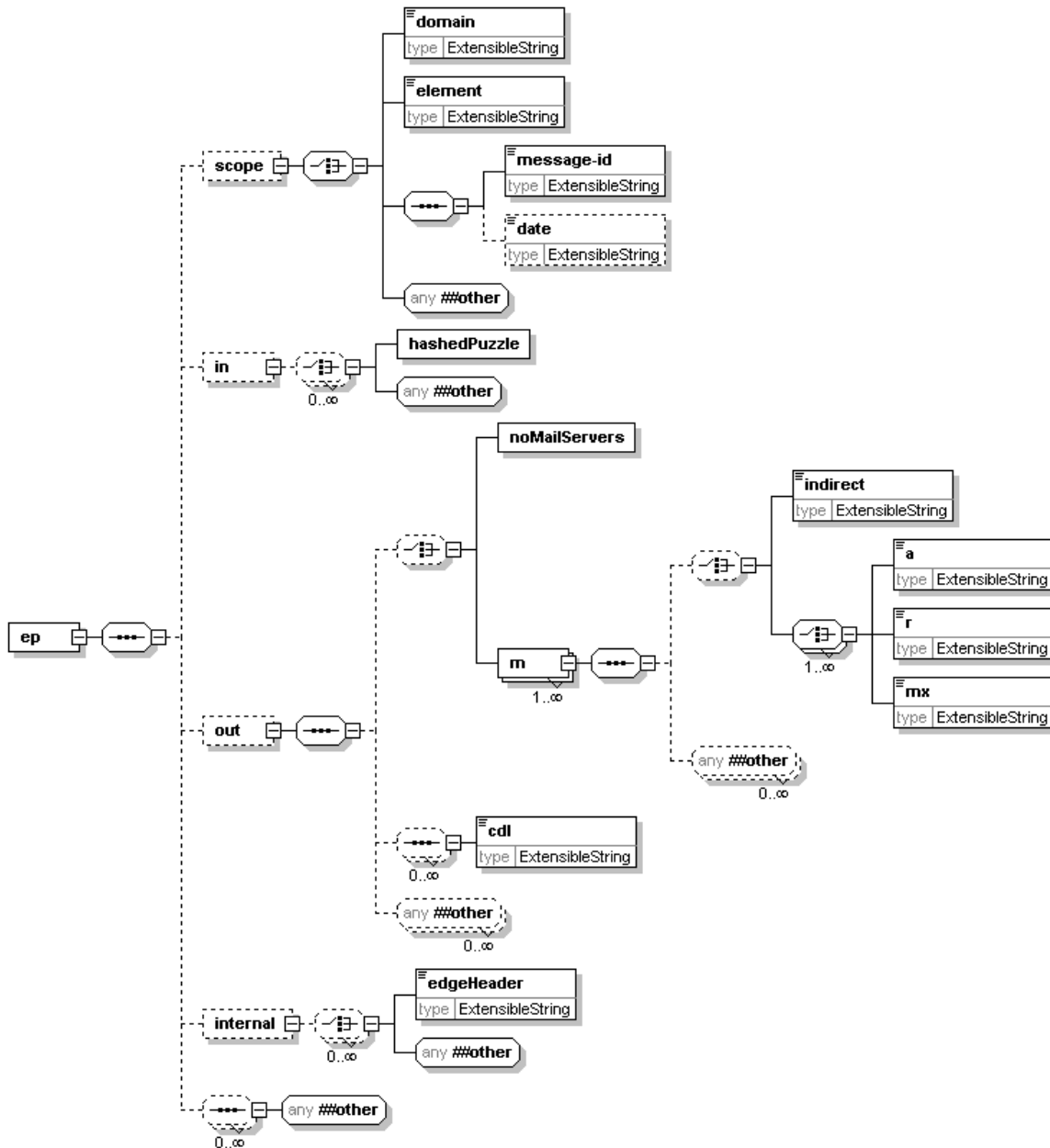
```

</xs:element>
<xs:element name="out" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:choice minOccurs="0">
        <xs:element name="noMailServers">
          <xs:complexType>
            <xs:anyAttribute namespace="##other" processContents="lax"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="m" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:choice minOccurs="0">
                <xs:element name="indirect" type="ExtensibleString"/>
                <xs:choice maxOccurs="unbounded">
                  <xs:element name="a" type="ExtensibleString"/>
                  <xs:element name="r" type="ExtensibleString"/>
                  <xs:element name="mx" type="ExtensibleString"/>
                </xs:choice>
              </xs:choice>
              <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="eNoop" type="xs:boolean" use="optional"/>
            <xs:attribute name="allETPSigned" type="xs:boolean" use="optional"/>
            <xs:anyAttribute namespace="##other" processContents="lax"/>
          </xs:complexType>
        </xs:element>
      </xs:choice>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="cdl" type="ExtensibleString"/>
      </xs:sequence>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="directOnly" type="xs:boolean" use="optional"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
</xs:element>
<xs:element name="internal" minOccurs="0">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="edgeHeader" type="ExtensibleString"/>
      <xs:any namespace="##other" processContents="lax"/>
    </xs:choice>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

```

```
</xs:element>
<xs:sequence minOccurs="0" maxOccurs="unbounded">
  <xs:any namespace="##other" processContents="lax"/>
</xs:sequence>
</xs:sequence>
<xs:attribute name="testing" type="xs:boolean" use="optional" default="false"/>
<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
</xs:element>
</xs:schema>
```

Pictorially, this schema can be represented as follows (note that XML attributes are not illustrated in this diagram; only XML elements are shown):



The `testing` attribute which may optionally be present on an `ep` element provides a means by which an E-mail Policy Document can be tentatively deployed in an experimental mode. Documents in which such attribute is present with a true value SHOULD be entirely ignored (one should act as if the document were absent) unless one has cause by some other means (such as a private arrangement with the document publisher) to do otherwise.

Most of the XML elements defined here allow for an extensible set of attributes to be attached; this is signified by the presence of definitions of the form `<xs:anyAttribute namespace="##other"/>` in the schema. Software interpreting an E-mail Policy Document SHOULD ignore such extended attributes that it finds which it doesn't semantically understand; those wishing to avail themselves of this mechanism MUST in their design allow that this may occur.

Similarly, several places in the schema permit extensibility in the form of the introduction of arbitrary new XML elements; this is signified by the presence in the schema of wildcard definitions of the form `<xs:any namespace="##other" processContents="lax"/>` and is illustrated in the schema diagrams above with the construct:



With one exception, software interpreting an E-mail Policy Document SHOULD simply ignore any extended elements that it finds which it doesn't semantically understand. The exception pertains to the extensible elements permitted in the `scope` element: E-mail Policy Documents indicated as applying to scopes not understood by the interpreting application should be ignored in their entirety. Those wishing to avail themselves of the use of extended XML elements in E-mail Policy Documents MUST in their design accommodate these processing rules.

The optional `scope` element indicates the applicability of this particular policy. The possible options are a given DNS domain, a particular e-mail address, or a particular e-mail message as designated by the value of its `Message-Id`: header; alternate scoping mechanisms can be accommodated in the wildcard child of the `scope` element. If the `scope` element is omitted, the scope it is to be inferred from contextual use. If an `ep` element is stored in the E-mail Policy Document of a given DNS domain, then the `scope` MUST either be present with a value of `domain` equal to the domain in question, or `scope` MUST be omitted.

Use of e-mail policy documents with `message-id` scope begins to enable a form of a machine-interpretable challenge-response mechanism. When attached to a transmitted message (in an `E-mailPolicy`: header) such a policy document indicates policy that the address listed in the `From`: header of the message has with regard to a second message, namely the one whose `Message-Id`: header is as indicated (and where, if present, the `scope/date` element, which must contain a date formatted per §3.6.1 of RFC2822, also semantically matches the `Date`: header on the message). On a challenging side, the policy indicated usually has its useful semantic content beneath the `ep/in` element, where it can list information that it wishes were known for the second identified message; on the response side, useful information is usually contained beneath the `ep/out` element. Further details of such mechanisms are beyond the scope of this specification.

The optional `internal` element indicates e-mail related policies that might be of interest internally within the scope (usually a domain) with which this policy is associated. Publishing this internal information in DNS might be a convenient mechanism for disseminating the information through the organization associated with the domain. Be aware, however, that generally speaking, such information will be publicly accessible, and so private or sensitive information should not be placed here. The one architected element within `internal` is `edgeHeader` which denotes, as previously described, a distinguished string which will reliably be found in `Received`: headers added the incoming SMTP serves on the edge of a given domain (in e-mail policy documents of non-domain scope, the `edgeHeader` element has no meaning). Publishing this information in this publicly accessible way should cause little concern, since the contents of the `Received`: headers in one's mail undoubtedly already make their way outside the organization as mail is forwarded, bounced, and so on, and so ought not to be sensitive information. That said, those organizations for which this remains a concern should simply not avail themselves of the `edgeHeader` mechanism.

The description of the remaining elements of this schema is beyond the scope of this specification.

Readers are reminded that XML Schema specifies that values of type boolean may use either "true" or "1" to represent a true value and either "false" or "0" to represent a false value.

The following is an example XML instance of an `ep` element that applies to the `example.com` domain. `example.com` list two outbound mail servers: the IP address 1.2.3.4, together with whatever IP addresses result from DNS address resolution on the domain `example.com` itself.

Note that, per reference [4], the encoding declaration `<?xml ... ?>` may generally be omitted on XML instances that are UTF-8 encoded.

```
<?xml version="1.0" encoding="UTF-8"?>
<ep xmlns='http://ms.net/1'>
  <scope>
    <domain>example.com</domain>
  </scope>
```

```
<out>
  <m>
    <a>1.2.3.4</a>
  </m>
  <m><mx/></m>
</out>
</ep>
```

5. References

- [1] Danisch, Hadmut, *The RMX DNS RR and method for lightweight SMTP sender authorization*, <http://www.danisch.de/work/security/txt/draft-danisch-dns-rr-smtp-03.txt>.
- [2] DeKok, A. (Ed.), *Lightweight MTA Authentication Protocol (LMAP) Discussion and Applicability Statement*, November 3, 2003, http://asrg.kavi.com/apps/group_public/download.php/18/draft-irtf-asrg-lmap-discussion-00.txt
- [3] Wong, Meng Weng, Hadmut Danish, Gordon Fecyk Mark Lentczner, *Sender Permitted From*, <http://spf.pobox.com>.
- [4] Worldwide Web Consortium, *Extensible Markup Language (XML) 1.0 (Second Edition)*, <http://www.w3.org/TR/REC-xml>

[⤴ Top of document](#)