

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 14, 2015

M. Kucherawy, Ed.
E. Zwicky, Ed.
Yahoo!
November 10, 2014

Domain-based Message Authentication, Reporting and Conformance (DMARC)
draft-kucherawy-dmarc-base-06

Abstract

Domain-based Message Authentication, Reporting and Conformance (DMARC) is a scalable mechanism by which a mail-originating organization can express domain-level policies and preferences for message validation, disposition, and reporting, and a mail receiving organization can use those policies and preferences to improve mail handling.

Originators of Internet Mail need to be able to associate reliable and authenticated domain identifiers with messages, communicate policies about messages that use those identifiers, and report about mail using those identifiers. These abilities have several benefits: Receivers can provide feedback to domain owners about the use of their domains, which can provide valuable insight about the management of internal operations and the presence of external domain name abuse.

DMARC does not produce or encourage elevated delivery privilege of authenticated email. DMARC is a mechanism for policy distribution that enables increasingly strict handling of messages that fail authentication checks, ranging from no action, through altered delivery, up to message rejection.

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 May 14, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](http://trustee.ietf.org/bcp78) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	5
2.	Requirements	6
2.1.	High-Level Goals	6
2.2.	Out Of Scope	7
2.3.	Scalability	7
2.4.	Anti-Phishing	8
3.	Terminology and Definitions	8
3.1.	Overview	9
3.2.	Organizational Domain	14
4.	Use of RFC5322.From	15
5.	Policy	16
5.1.	DMARC Policy Record	16
5.2.	DMARC URIs	17
5.3.	General Record Format	17
5.4.	Formal Definition	21
5.5.	Domain Owner Actions	22
5.6.	Mail Receiver Actions	23
5.7.	Policy Enforcement Considerations	26
6.	DMARC Feedback	27
6.1.	Verifying External Destinations	28
6.2.	Aggregate Reports	29
6.3.	Failure Reports	35
7.	Minimum Implementations	37
8.	Privacy Considerations	37
8.1.	Data Exposure Considerations	37
8.2.	Report Recipients	38
8.3.	Secure Protocols	39
9.	Other Topics	39
9.1.	Issues Specific to SPF	39
9.2.	DNS Load and Caching	39
9.3.	Rejecting Messages	40
9.4.	Identifier Alignment Considerations	41
10.	IANA Considerations	41
10.1.	Authentication-Results Method Registry Update	41
10.2.	Authentication-Results Result Registry Update	41
10.3.	Feedback Report Header Fields Registry Update	43
10.4.	DMARC Tag Registry	43
10.5.	DMARC Report Format Registry	44
11.	Security Considerations	45
11.1.	Authentication Methods	45
11.2.	Attacks on Reporting URIs	45
11.3.	DNS Security	45
11.4.	Display Name Attacks	46
11.5.	External Reporting Addresses	46
12.	References	47
12.1.	Normative References	47

12.2.	Informative References	48
Appendix A.	Technology Considerations	50
A.1.	S/MIME	50
A.2.	Method Exclusion	50
A.3.	Sender Header Field	51
A.4.	Domain Existence Test	52
A.5.	Issues With ADSP In Operation	52
A.6.	Organizational Domain Discovery Issues	53
Appendix B.	Examples	54
B.1.	Identifier Alignment examples	54
B.2.	Domain Owner example	56
B.3.	Mail Receiver Example	60
B.4.	Utilization of Aggregate Feedback example	62
B.5.	mailto Transport example	63
Appendix C.	DMARC XML Schema	64
Appendix D.	Public Discussion	70
Appendix E.	Acknowledgements	70
	Authors' Addresses	70

1. Introduction

The Sender Policy Framework ([\[SPF\]](#)) and DomainKeys Identified Mail ([\[DKIM\]](#)) provide domain-level authentication. They enable cooperating email receivers to detect mail authorized to use the domain name, which can permit differential handling. (A detailed discussion of the threats these systems attempt to address can be found in [\[DKIM-THREATS\]](#).) However, there has been no single widely accepted or publicly available mechanism to communication of domain-specific message handling policies for receivers, or to request reporting of authentication and disposition of received mail. Absent the ability to obtain feedback reports, originators who have implemented email authentication have difficulty determining how effective their authentication is. As a consequence, use of authentication failures to filter mail typically does not succeed.

Over time, one-on-one relationships were established between select senders and receivers with privately communicated means to assert policy and receive message traffic and authentication disposition reporting. Although these ad hoc practices have been generally successful, they require significant manual coordination between parties, and this model does not scale for general use on the Internet.

This document defines Domain-based Message Authentication, Reporting and Compliance (DMARC), a mechanism by which email operators leverage existing authentication and policy advertisement technologies to enable both message-stream feedback and enforcement of policies against unauthenticated email.

DMARC allows domain owners and receivers to collaborate by:

1. Providing receivers with assertions about domain owners' policies
2. Providing feedback to senders so they can monitor authentication and judge threats

The basic outline of DMARC is:

1. Domain owners publish policy assertions about domains via the DNS.
2. Receivers compare the [RFC5322](#) From: address in the mail to the SPF and DKIM results, if present, and the DMARC policy in DNS.
3. These receivers can use these results to determine how the mail should be handled.

4. The receiver reports to the domain owner about mail claiming to be from their domain.

Security terms used in this document are defined in [[SEC-TERMS](#)].

DMARC differs from previous approaches to policy advertisement (e.g., [[SPF](#)] and [[ADSP](#)]) in that:

- o Authentication technologies are:
 1. decoupled from any technology-specific policy mechanisms; and
 2. used solely to establish reliable per-message domain-level identifiers.
- o Multiple authentication technologies are used to:
 1. reduce the impact of transient authentication errors
 2. reduce the impact of site-specific configuration errors and deployment gaps
 3. enable more use cases than any individual technology supports alone
- o Receiver-generated feedback is supported, allowing senders to establish confidence in authentication practices.
- o The domain name extracted from a message's [RFC5322](#).From field is the primary identifier in the DMARC mechanism. This identifier is used in conjunction with the results of the underlying authentication technologies to evaluate results under DMARC.

[2.](#) Requirements

Specification of DMARC is guided by the following high-level goals, security dependencies, detailed requirements, and items that are documented as out-of-scope.

[2.1.](#) High-Level Goals

DMARC has the following high-level goals:

- o Allow Domain Owners to assert the preferred handling of authentication failures, for messages purporting to have authorship within the domain.

- o Allow Domain Owners to verify their authentication deployment.
- o Minimize implementation complexity for both senders and receivers, as well as the impact on handling and delivery of legitimate messages.
- o Reduce the amount of successfully delivered spoofed email.
- o Work at Internet scale.

2.2. Out Of Scope

Several topics and issues are specifically out of scope for the initial version of this work. This includes the following:

- o different treatment of messages that are not authenticated versus those that fail authentication;
- o evaluation of anything other than [RFC5322](#).From;
- o multiple reporting formats;
- o publishing policy other than via the DNS;
- o reporting or otherwise evaluating other than the last-hop IP address;
- o attacks in the [RFC5322](#).From field, also known as "display-name" attacks;
- o authentication of entities other than domains, since DMARC is built upon SPF and DKIM which authenticate domains; and
- o content analysis.

2.3. Scalability

Scalability is a major issue for systems that need to operate in a system as widely deployed as current SMTP email. For this reason, DMARC seeks to avoid the introduction of third parties or pre-sending agreements between senders and receivers. This preserves the positive aspects of the current email infrastructure.

Although DMARC does not introduce third parties to the email handling flow, it also does not preclude them. Third parties are free to provide services in conjunction with DMARC.

2.4. Anti-Phishing

DMARC is designed to prevent bad actors from sending mail that claims to come from legitimate senders, particularly senders of transactional email (official mail that is about business transactions). One of the primary uses of this kind of spoofed mail is phishing (enticing users to provide information by pretending to be the legitimate service requesting the information). Thus, DMARC is significantly informed by ongoing efforts to enact large-scale, Internet-wide, anti-phishing measures.

Although DMARC can only be used to combat specific forms of exact-domain spoofing directly, the DMARC mechanism has been found to be useful in the creation of reliable and defensible message streams.

DMARC does not attempt to solve all problems with spoofed or otherwise fraudulent email. In particular, it does not address the use of visually similar domain names ("cousin domains") or abuse of the [RFC5322](#) [MAIL].From human readable <display-name>.

3. Terminology and Definitions

This section defines terms used in the rest of the document.

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

Readers are encouraged to be familiar with the contents of [EMAIL-ARCH]. In particular, that document defines various roles in the messaging infrastructure that can appear the same or separate in various contexts. For example, a Domain Owner could, via the messaging security mechanisms on which DMARC is based, delegate the ability to send mail as the Domain Owner to a third party with another role. This document does not address the distinctions among such roles; the reader is encouraged to become familiar with that material before continuing.

The following terms are also used:

Authenticated Identifiers: Domain-level identifiers that are validated using authentication technologies are referred to as "Authenticated Identifiers". See [Section 3.1.1](#) for details about the supported mechanisms.

Author Domain: The domain name of the apparent author, as extracted from the [RFC5322](#).From field.

Domain Owner: An entity or organization that owns a DNS domain. The term "owns" here indicates that the entity or organization being referenced holds the registration of that DNS domain. Domain Owners range from complex, globally-distributed organizations, to service providers working on behalf of non-technical clients, to individuals responsible for maintaining personal domains. This specification uses this term as analogous to an Administrative Management Domain as defined in [\[EMAIL-ARCH\]](#).

Identifier Alignment: When the domain in the [RFC5322](#).From address matches a domain validated by SPF or DKIM (or both), it has Identifier Alignment.

Mail Receiver: The entity or organization that receives and processes email. Mail Receivers operate one or more Internet-facing Mail Transport Agents (MTAs).

Organizational Domain: The domain that was registered with a domain name registrar. In the absence of more accurate methods, heuristics are used to determine this, since it is not always the case that the registered domain name is simply a top-level DNS domain plus one component (e.g., "example.com", where "com" is a top-level domain). The Organizational Domain is determined by applying the algorithm found in [Section 3.2](#).

Report Receiver: An operator that receives reports from another operator implementing the reporting mechanism described in this document. Such an operator might be receiving reports about its own messages, or reports about the messages claiming to be from a third party. This term applies collectively to the system components that receive and process these reports and the organizations that operate them.

[3.1](#). Overview

This section provides a general overview of the design and operation of the DMARC environment.

[3.1.1](#). Authentication Mechanisms

The following mechanisms for determining Authenticated Identifiers are supported in this version of DMARC:

- o [\[DKIM\]](#), which provides a domain-level identifier in the content of the "d=" tag of a validated DKIM-Signature header field.

- o [\[SPF\]](#), which authenticates the domain found in an [\[SMTP\]](#) MAIL command when it is the authorized domain.

[3.1.2.](#) Key Concepts

DMARC policies are published by the Domain Owner, and retrieved by the Mail Receiver during the SMTP session, via the DNS.

DMARC's filtering function is based on whether SPF or DKIM can provide an authenticated, aligned identifier for the message under consideration. Messages that purport to be from a Domain Owner's domain and arrive from servers that are not authorized by SPF and do not contain an appropriate DKIM signature can be affected by DMARC policies.

It is important to note that the authentication mechanisms employed by DMARC authenticate only a DNS domain, and do not authenticate the local-part of any email address identifier found in a message, nor does it validate the legitimacy of message content.

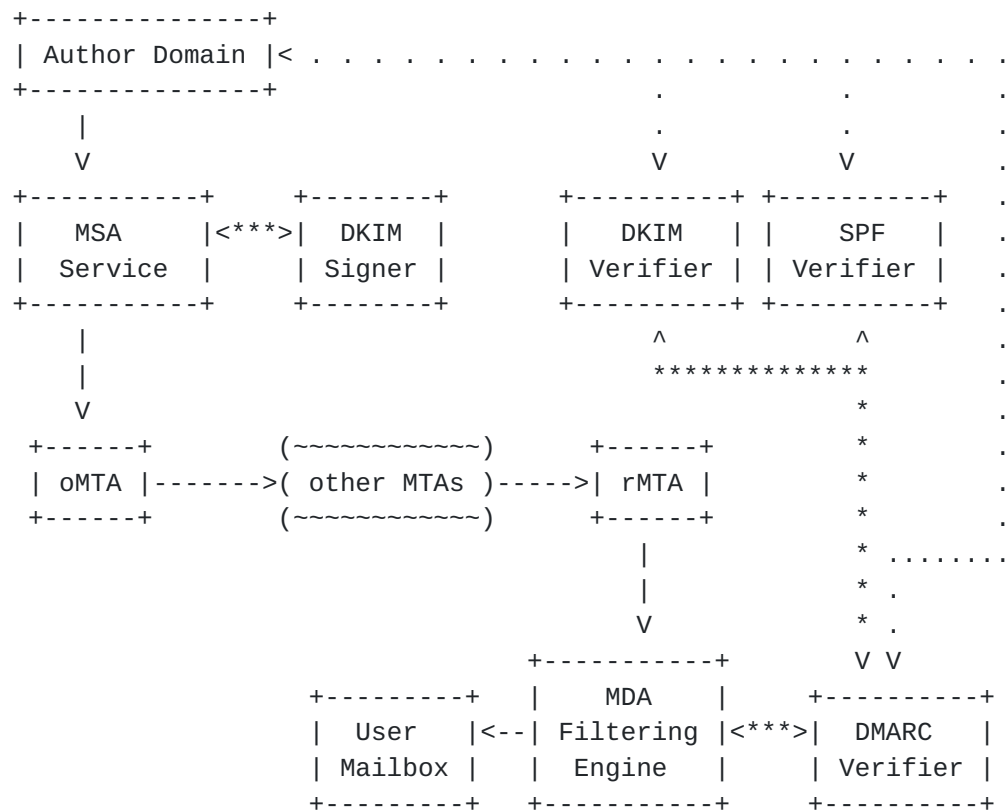
DMARC's feedback component involves the collection of information about received messages claiming to be from the Organizational Domain for periodic aggregate reports to the Domain Owner. The parameters and format for such reports are discussed in later sections of this document.

A DMARC-enabled Mail Receiver might also generate per-message reports that contain information related to individual messages that fail SPF and/or DKIM. Per-message failure reports are a useful source of information when debugging deployments (if messages can be determined to be legitimate even though failing authentication) or in analyzing attacks. The capability for such services is enabled by DMARC but defined in other referenced material.

A message satisfies the DMARC checks if at least one of the supported authentication mechanisms:

1. produces a "pass" result; and
2. produces that result based on an identifier that is in alignment, as defined in [Section 3](#).

[3.1.3.](#) Flow Diagram



The above diagram shows a simple flow of messages through a DMARC-aware system. Solid lines denote the actual message flow, dotted lines involve Domain Name System queries used to retrieve message policy related to the supported message authentication schemes, and asterisk lines indicate data exchange between message handling modules and message authentication modules. "sMTA" is the sending MTA, and "rMTA" is the receiving MTA.

In essence the steps are as follows:

1. Author Domain constructs an SPF policy and publishes it in its DNS database as per [SPF]. Author Domain also configures its system for DKIM signing as described in [DKIM]. Finally, Author Domain publishes via the DNS a DMARC message handling policy.
2. Author generates a message and hands the message to Author Domain's designated mail submission service.
3. Submission service passes relevant details to the DKIM signing module in order to generate a DKIM signature to be applied to the message.
4. Submission service relays the now-signed message to its designated transport service for routing to its intended

recipient(s).

5. Message may pass through other relays, but eventually arrives at a recipient's transport service.
6. Recipient delivery service conducts SPF and DKIM authentication checks by passing the necessary data to their respective modules, each of which require queries to the Author Domain's DNS data (when identifiers are aligned; see below).
7. The results of these are passed to the DMARC module along with the Author domain. The DMARC module attempts to retrieve a policy from the DNS for that domain. If none is found, the DMARC module determines the Organizational Domain and repeats the attempt to retrieve a policy from the DNS. (This is described in further detail in [Section 5.6.3.](#))
8. If a policy is found, it is combined with the Author Domain and the SPF and DKIM results to produce a DMARC policy result (e.g., a "pass" or "fail"), and can optionally cause one of two kinds of reports to be generated (not shown).
9. Recipient transport service either delivers the message to the recipient inbox, or takes other local policy action based on the DMARC result (not shown).

3.1.4. Identifier Alignment

Email authentication technologies authenticate various (and disparate) aspects of an individual message. For example, [\[DKIM\]](#) authenticates the domain that affixed a signature to the message, while [\[SPF\]](#) authenticates either the domain that appears in the [RFC5321](#).MailFrom portion of [\[SMTP\]](#) or the [RFC5321](#).EHLO/HELO domain if the [RFC5321](#).MailFrom is null (in the case of Delivery Status Notifications). These may be different domains, and they are typically not visible to the end user.

DMARC uses the [RFC5322](#) [\[MAIL\]](#).From domain to evaluate the applicability of Authenticated Identifiers. The [RFC5322](#) [\[MAIL\]](#).From domain was selected as the central identity of the DMARC mechanism because it is a required message header field and therefore guaranteed to be present in compliant messages, and most MUAs represent the [RFC5322](#) [\[MAIL\]](#).From field as the originator of the message and render some or all of this header field's content to end users.

Thus, this field is the one used by end users to identify the source of the message, and therefore is a prime target for abuse. Many

high-profile email sources, such as email service providers, require that the sending agent have authenticated before email can be generated. Thus, for these mailboxes, the mechanism described in this document provides recipient end users with strong evidence that the message was indeed originated by the agent they associate with that mailbox, if the end user knows that these various protections have been provided.

Domain names in this context are to be compared in a case-insensitive manner, per [\[DNS-CASE\]](#).

It is important to note that identifier alignment cannot occur with a message that is not valid per [\[MAIL\]](#), particularly one with a malformed, absent, or repeated [RFC5322.From](#) field, since in that case there is no reliable way to determine a DMARC policy that applies to the message. Accordingly, DMARC operation is predicated on the input being a valid [RFC5322](#) message object, and handling of such non-compliant cases is outside of the scope of this specification. Further discussion of this can be found in [Section 5.6.1](#).

Each of the underlying authentication technologies that DMARC takes as input yield authenticated domains as their outputs when they succeed. From the perspective of DMARC, each can be operated in a "strict" mode or a "relaxed" mode. A Domain Owner would normally select "strict" mode if it wanted Mail Receivers to apply DMARC processing only to messages bearing an [RFC5322.From](#) domain exactly matching the domains those mechanisms will verify. Using "relaxed" mode can be used when the operator also wishes to affect message flows bearing subdomains of the verified domains.

[3.1.4.1](#). DKIM-authenticated Identifiers

DMARC provides the option of applying DKIM in a strict or relaxed identifier alignment mode. (Note that these are not related to DKIM's "simple" and "relaxed" canonicalization modes.)

In relaxed mode, the Organizational Domains of both the [\[DKIM\]](#)-authenticated signing domain (taken from the value of the "d=" tag in the signature) and that of the [RFC5322.From](#) domain must be equal if the identifiers are to be considered aligned. In strict mode, only an exact match between both of the Fully Qualified Domain Names (FQDN) considered to produce identifier alignment.

To illustrate, in relaxed mode, if a validated DKIM signature successfully verifies with a "d=" domain of "example.com", and the [RFC5322.From](#) address is "alerts@news.example.com", the DKIM "d=" domain and the [RFC5322.From](#) domain are considered to be "in alignment". In strict mode, this test would fail since the "d="

domain does not exactly match the FQDN of the address.

However, a DKIM signature bearing a value of "d=com" would never allow an "in alignment" result as "com" should appear on all public suffix lists, and therefore cannot be an Organizational Domain.

Identifier alignment is required because a message can bear a valid signature from any domain, including domains used by a mailing list or even a bad actor. Therefore, merely bearing a valid signature is not enough to infer authenticity of the Author Domain.

Note that a single email can contain multiple DKIM signatures, and it is considered to be a DMARC "pass" if any DKIM signature is aligned and verifies.

3.1.4.2. SPF-authenticated Identifiers

DMARC provides the option of applying SPF in a strict or relaxed identifier alignment mode.

In relaxed mode, the [[SPF](#)]-authenticated domain and [RFC5322](#).From domain must have the same Organizational Domain. In strict mode, only an exact DNS domain match is considered to produce identifier alignment.

For example, if a message passes an SPF check with an [RFC5321](#).MailFrom domain of "cbg.bounces.example.com", and the address portion of the [RFC5322](#).From field contains "payments@example.com", the Authenticated [RFC5321](#).MailFrom domain identifier and the [RFC5322](#).From domain are considered to be "in alignment" in relaxed mode, but not in strict mode.

3.1.4.3. Alignment and Extension Technologies

If in the future DMARC is extended to include the use of other authentication mechanisms, the extensions will need to allow for domain identifier extraction so that alignment with the [RFC5322](#).From domain can be verified.

3.2. Organizational Domain

The Organizational Domain is determined using the following algorithm:

1. Acquire a "public suffix" list, i.e., a list of DNS domain names reserved for registrations. Some country TLDs make specific registration requirements, e.g. the United Kingdom places company registrations under ".co.uk"; other TLDs such as ".com" appear in

the IANA registry of top-level DNS domains. A public suffix list is the union of all of these. [Appendix A.6.1](#) contains some discussion about obtaining a public suffix list.

2. Break the subject DNS domain name into a set of "n" ordered labels. Number these labels from right-to-left; e.g. for "example.com", "com" would be label 1 and "example" would be label 2.
3. Search the public suffix list for the name that matches the largest number of labels found in the subject DNS domain. Let that number be "x".
4. Construct a new DNS domain name using the name that matched from the public suffix list and prefixing to it the "x+1"th label from the subject domain. This new name is the Organizational Domain.

Thus, since "com" is an IANA-registered TLD, a subject domain of "a.b.c.d.example.com" would have an Organizational Domain of "example.com".

The process of determining a suffix is currently a heuristic one. No list is guaranteed to be accurate or current.

4. Use of [RFC5322.From](#)

One of the most obvious points of security scrutiny for DMARC is the choice to focus on an identifier, namely the [RFC5322](#).From address, which is part of a body of data that has been trivially forged throughout the history of email.

Several points suggest it is the most correct and safest thing to do in this context:

- o Of all the identifiers that are part of the message itself, this is the only one guaranteed to be present.
- o It seems the best choice of an identifier on which to focus as most MUAs display some or all of the contents of that field in a manner strongly suggesting those data as reflective of the true originator of the message.

The absence of a single, properly-formed [RFC5322.From](#) field renders the message invalid. Handling of such a message is outside of the scope of this specification.

Since the sorts of mail typically protected by DMARC participants

tend to only have single Authors, DMARC participants generally operate under a slightly restricted profile of [RFC5322](#) with respect to the expected syntax of this field. See [Section 5.6](#) for details.

5. Policy

DMARC policies are published by Domain Owners and applied by Mail Receivers.

A Domain Owner advertises DMARC participation of one or more of its domains by adding a DNS TXT record (described in [Section 5.1](#)) to those domains. In doing so, Domain Owners make specific requests of Mail Receivers regarding the disposition of messages purporting to be from one of the Domain Owner's domains and the provision of feedback about those messages.

A Domain Owner may choose not to participate in DMARC evaluation by Mail Receivers. In this case, the Domain Owner simply declines to advertise participation in those schemes. For example, if the results of path authorization checks ought not be considered as part of the overall DMARC result for a given Author Domain, then the Domain Owner does not publish an SPF policy record that can produce an SPF pass result.

A Mail Receiver implementing the DMARC mechanism SHOULD make a best-effort attempt to adhere to the Domain Owner's published DMARC policy when a message fails the DMARC test. Since email streams can be complicated (due to forwarding, existing [RFC5322](#).From domain-spoofing services, etc.), Mail Receivers MAY deviate from a Domain Owner's published policy during message processing and SHOULD make available the fact of and reason for the deviation to the Domain Owner via feedback reporting, specifically using the "PolicyOverride" feature of the aggregate report (see [Section 6.2](#)).

5.1. DMARC Policy Record

Domain Owner DMARC preferences are stored as DNS TXT records in subdomains named "_dmarc". For example, the Domain Owner of "example.com" would post DMARC preferences in a TXT record at "_dmarc.example.com". Similarly, a Mail Receiver wishing to query for DMARC preferences regarding mail with an [RFC5322](#).From domain of "example.com" would issue a TXT query to the DNS for the subdomain of "_dmarc.example.com". The DNS-located DMARC preference data will hereafter be called the "DMARC record".

DMARC's use of the Domain Name Service is driven by DMARC's use of domain names and the nature of the query it performs. The query

requirement matches perfectly with the DNS, for obtaining simple parametric information. It uses an established method of storing the information, associated with the target domain name, namely an isolated TXT record that is restricted to the DMARC context. Use of the DNS as the query service has the considerable benefit of re-using an extremely well-established operations, administration and management infrastructure, rather than creating a new one.

Per [\[DNS\]](#), a TXT record can comprise several "character-string" objects. Where this is the case, the module performing DMARC evaluation MUST concatenate these strings by joining together the objects in order and parsing the result as a single string.

5.2. DMARC URIs

[\[URI\]](#) defines a generic syntax for identifying a resource. The DMARC mechanism uses this as the format by which a Domain Owner specifies the destination for the two report types (RUA and RUF) that are supported.

The place such URIs are specified (see [Section 5.3](#)) allows a list of these to be provided. A report is normally sent to each listed URI in the order provided by the Domain Owner. Receivers MAY impose a limit on the number of URIs to which they will send reports, but MUST support the ability to send to at least two. The list of URIs is separated by commas (ASCII 0x2C).

Each URI can have associated with it a maximum report size that may be sent to it. This is accomplished by appending an exclamation point (ASCII 0x21), followed by a maximum size indication, before a separating comma or terminating semi-colon.

Thus, a DMARC URI is a URI within which any commas or exclamation points are percent-encoded per [\[URI\]](#), followed by an OPTIONAL exclamation point and a maximum size specification, and, if there are additional reporting URIs in the list, a comma and the next URI.

For example, the URI "mailto:reports@example.com!50m" would request a report be sent via email to "reports@example.com" so long as the report payload does not exceed 50 megabytes.

A formal definition is provided in [Section 5.4](#).

5.3. General Record Format

DMARC records follow the extensible "tag-value" syntax for DNS-based key records defined in DKIM [\[DKIM\]](#).

[Section 10](#) creates a registry for known DMARC tags and registers the initial set defined in this document. Only tags defined in this document or in later extensions, and thus added to that registry, are to be processed; unknown tags MUST be ignored.

The following tags are introduced as the initial valid DMARC tags:

adkim: (plain-text; OPTIONAL, default is "r".) Indicates whether strict or relaxed DKIM identifier alignment mode is required by the Domain Owner. See [Section 3.1.4.1](#) for details.

aspf: (plain-text; OPTIONAL, default is "r".) Indicates whether strict or relaxed SPF identifier alignment mode is required by the Domain Owner. See [Section 3.1.4.2](#) for details.

fo: Failure reporting options (plain-text; OPTIONAL, default "0")
Provides requested options for generation of failure reports. Report generators MAY choose to adhere to the requested options. This tag's content MUST be ignored if a "ruf" tag (below) is not also specified. The value of this tag is a colon-separated list of characters that indicate failure reporting options as follows:

0: Generate a DMARC failure report if all underlying authentication mechanisms failed to produce an aligned "pass" result.

1: Generate a DMARC failure report if any underlying authentication mechanism produced something other than an aligned "pass" result.

d: Generate a DKIM failure report if the message had a signature that failed evaluation, regardless of its alignment. DKIM-specific reporting is described in [\[AFRF-DKIM\]](#).

s: Generate an SPF failure report if the message failed SPF evaluation, regardless of its alignment. SPF-specific reporting is described in [\[AFRF-SPF\]](#).

p: Requested Mail Receiver policy (plain-text; REQUIRED for policy records). Indicates the policy to be enacted by the Receiver at the request of the Domain Owner. Policy applies to the domain queried and to sub-domains unless sub-domain policy is explicitly described using the "sp" tag. This tag is mandatory for policy records only, but not for third-party reporting records (see [Section 6.1](#)). Possible values are as follows:

none: The Domain Owner requests no specific action be taken regarding delivery of messages.

quarantine: The Domain Owner wishes to have email that fails the DMARC mechanism check to be treated by Mail Receivers as suspicious. Depending on the capabilities of the Mail Receiver, this can mean "place into spam folder", "scrutinize with additional intensity", and/or "flag as suspicious".

reject: The Domain Owner wishes for Mail Receivers to reject email that fails the DMARC mechanism check. Rejection SHOULD occur during the SMTP transaction. See [Section 9.3](#) for some discussion of SMTP rejection methods and their implications.

pct: (plain-text integer between 0 and 100, inclusive; OPTIONAL; default is 100). Percentage of messages from the Domain Owner's mail stream to which the DMARC mechanism is to be applied. However, this MUST NOT be applied to the DMARC-generated reports, all of which must be sent and received unhindered. The purpose of the "pct" tag is to allow Domain Owners to enact a slow rollout enforcement of the DMARC mechanism. The prospect of "all or nothing" is recognized as preventing many organizations from experimenting with strong authentication-based mechanisms. See [Section 5.6.4](#) for details. Note that random selection based on this percentage, such as the following pseudocode, is adequate:

```
if (random mod 100) < pct then
    selected = true
else
    selected = false
```

rf: Format to be used for message-specific failure reports (colon-separated plain-text list of values; OPTIONAL; default "afrf"). The value of this tag is a list of one or more report formats as requested by the Domain Owner to be used when a message fails both [\[SPF\]](#) and [\[DKIM\]](#) tests to report details of the individual failure. The values MUST be present in the registry of reporting formats defined in [Section 10](#); a Mail Receiver observing a different value SHOULD ignore it, or MAY ignore the entire DMARC record. For this version, only "afrf" (defined in [\[AFRF\]](#)) is presently supported. See [Section 6.3](#) for details. For interoperability, the AFRF format MUST be supported.

ri: Interval requested between aggregate reports (plain-text, 32-bit unsigned integer; OPTIONAL; default 86400). Indicates a request to Receivers to generate aggregate reports separated by no more than the requested number of seconds. DMARC implementations MUST be able to provide daily reports and SHOULD be able to provide

hourly reports when requested. However, anything other than a daily report is understood to be accommodated on a best-effort basis.

rua: Addresses to which aggregate feedback is to be sent (comma-separated plain-text list of DMARC URIs; OPTIONAL). A comma or exclamation point that is part of such a DMARC URI MUST be encoded per Section 2.1 of [[URI](#)] so as to distinguish it from the list delimiter or an OPTIONAL size limit. [Section 6.1](#) discusses considerations that apply when the domain name of a URI differs from that of the domain advertising the policy. See [Section 11.5](#) for additional considerations. Any valid URI can be specified. A Mail Receiver MUST implement support for a "mailto:" URI, i.e. the ability to send a DMARC report via electronic mail. If not provided, Mail Receivers MUST NOT generate aggregate feedback reports. URIs not supported by Mail Receivers MUST be ignored. The aggregate feedback report format is described in [Section 6.2](#).

ruf: Addresses to which message-specific failure information is to be reported (comma-separated plain-text list of DMARC URIs; OPTIONAL). If present, the Domain Owner is requesting Mail Receivers to send detailed failure reports about messages that fail the DMARC evaluation in specific ways (see the "fo" tag above). The format of the message to be generated MUST follow that specified in the "rf" tag. [Section 6.1](#) discusses considerations that apply when the domain name of a URI differs from that of the domain advertising the policy. A Mail Receiver MUST implement support for a "mailto:" URI, i.e. the ability to send a DMARC report via electronic mail. If not provided, Mail Receivers MUST NOT generate failure reports. See [Section 11.5](#) for additional considerations.

sp: Requested Mail Receiver policy for all subdomains (plain-text; OPTIONAL). Indicates the policy to be enacted by the Receiver at the request of the Domain Owner. It applies only to subdomains of the domain queried and not to the domain itself. Its syntax is identical to that of the "p" tag defined above. If absent, the policy specified by the "p" tag MUST be applied for subdomains. Note that "sp" will be ignored for DMARC records published on subdomains of Organizational Domains due to the effect of the DMARC Policy Discovery mechanism described in [Section 5.6.3](#).

v: Version (plain-text; REQUIRED). Identifies the record retrieved as a DMARC record. It MUST have the value of "DMARC1". The value of this tag MUST match precisely; if it does not or it is absent, the entire retrieved record MUST be ignored. It MUST be the first tag in the list.

A DMARC policy record MUST comply with the formal specification found in [Section 5.4](#) in that the "v" and "p" tags MUST be present and MUST appear in that order. Unknown tags MUST be ignored. Syntax errors in the remainder of the record SHOULD be discarded in favor of default values (if any) or ignored outright.

Note that given the rules of the previous paragraph, addition of a new tag into the registered list of tags does not itself require a new version of DMARC to be generated (with a corresponding change to the "v" tag's value), but a change to any existing tags does require a new version of DMARC.

[5.4.](#) Formal Definition

The formal definition of the DMARC format using [\[ABNF\]](#) is as follows:

```
dmARC-uri      = URI [ "!" 1*DIGIT [ "k" / "m" / "g" / "t" ] ]
                  ; "URI" is imported from \[URI\]; commas (ASCII
                  ; 0x2c) and exclamation points (ASCII 0x21)
                  ; MUST be encoded; the numeric portion MUST fit
                  ; within an unsigned 64-bit integer

dmARC-record   = dmARC-version dmARC-sep
                  [dmARC-request]
                  [dmARC-sep dmARC-srequest]
                  [dmARC-sep dmARC-auri]
                  [dmARC-sep dmARC-furi]
                  [dmARC-sep dmARC-adkim]
                  [dmARC-sep dmARC-aspf]
                  [dmARC-sep dmARC-ainterval]
                  [dmARC-sep dmARC-fo]
                  [dmARC-sep dmARC-rfmt]
                  [dmARC-sep dmARC-percent]
                  [dmARC-sep]
                  ; components other than dmARC-version and
                  ; dmARC-request may appear in any order

dmARC-version  = "v" *WSP "=" *WSP %x44 %x4d %x41 %x52 %x43 %x31

dmARC-sep      = *WSP %3b *WSP

dmARC-request  = "p" *WSP "=" *WSP
                  ( "none" / "quarantine" / "reject" )

dmARC-srequest = "sp" *WSP "=" *WSP
                  ( "none" / "quarantine" / "reject" )

dmARC-auri     = "rua" *WSP "=" *WSP
```



```
        dmarc-uri *(*WSP "," *WSP dmarc-uri)

dmarc-furi      = "ruf" *WSP "=" *WSP
                  dmarc-uri *(*WSP "," *WSP dmarc-uri)

dmarc-adkim     = "adkim" *WSP "=" *WSP
                  ( "r" / "s" )

dmarc-aspf      = "aspf" *WSP "=" *WSP
                  ( "r" / "s" )

dmarc-ainterval = "ri" *WSP "=" *WSP 1*DIGIT

dmarc-fo        = "fo" *WSP "=" *WSP
                  ( "0" / "1" / "d" / "s" )
                  *(*WSP ":" *WSP ( "0" / "1" / "d" / "s" ))

dmarc-rfmt      = "rf" *WSP "=" *WSP
                  ( "afrf" / "iodef" )

dmarc-percent   = "pct" *WSP "=" *WSP
                  1*3DIGIT
```

A size limitation in a dmarc-uri, if provided, is interpreted as a count of units followed by an OPTIONAL unit size ("k" for kilobytes, "m" for megabytes, "g" for gigabytes, "t" for terabytes). Without a unit, the number is presumed to be a basic byte count. Note that the units are considered to be powers of two; a kilobyte is 2¹⁰, a megabyte is 2²⁰, etc.

5.5. Domain Owner Actions

To implement the DMARC mechanism, the only action required of a Domain Owner is the creation of the DMARC policy record in the DNS. However, in order to make meaningful use of DMARC, a Domain Owner must at minimum either establish an address to receive reports, or deploy authentication technologies and ensure identifier alignment. Most Domain Owners will want to do both.

DMARC reports will be of significant size and the addresses that receive them are publicly visible, so we encourage Domain Owners to set up dedicated email addresses to receive and process reports, and to deploy abuse countermeasures on those email addresses as appropriate.

Authentication technologies are discussed in [\[DKIM\]](#) (see also [\[DKIM-OVERVIEW\]](#) and [\[DKIM-DEPLOYMENT\]](#)) and [\[SPF\]](#).

5.6. Mail Receiver Actions

This section describes receiver actions in the DMARC environment.

5.6.1. Extract Author Domain

The domain in the [RFC5322.From](#) field is extracted as the domain to be evaluated by DMARC. If the domain is encoded with UTF-8, the domain name must be converted to an A-label, as described in Section 2.3 of [\[IDNA\]](#), for further processing.

In order to be processed by DMARC, a message typically needs to contain exactly one [RFC5322.From](#): domain (a single From: field with a single domain in it). Not all messages meet this requirement, and handling of them is outside of the scope of this document. Typical exceptions, and the way they have been historically handled by DMARC participants, are as follows:

- o Messages with multiple [RFC5322.From](#) fields are typically rejected since that form is forbidden under [RFC 5322 \[MAIL\]](#);
- o Messages bearing a single [RFC5322.From](#) field containing multiple addresses (and, thus, multiple domain names to be evaluated) are typically rejected because the sorts of mail normally protected by DMARC do not use this format;
- o Messages that have no [RFC5322.From](#) field at all are typically rejected since that form is forbidden under [RFC 5322 \[MAIL\]](#);
- o Messages with an [RFC5322.From](#) field that contains no meaningful domains, such as [RFC 5322 \[MAIL\]](#)'s "group" syntax are typically ignored.

The case of a syntactically valid multi-valued [RFC5322.From](#) field presents a particular challenge. The process in this case is to apply the DMARC check using each of those domains found in the [RFC5322.From](#) field as the Author Domain, and apply the most strict policy selected among the checks that fail.

5.6.2. Determine Handling Policy

To arrive at a policy for an individual message, Mail Receivers MUST perform the following actions or their semantic equivalents. Steps 2-4 MAY be done in parallel, whereas steps 5 and 6 require input from previous steps.

The steps are as follows:

1. Extract the [RFC5322.From](#) domain from the message (as above).
2. Query the DNS for a DMARC policy record. Continue if one is found, or terminate DMARC evaluation otherwise. See [Section 5.6.3](#) for details.
3. Perform DKIM signature verification checks. A single email could contain multiple DKIM signatures. The results of this step are passed to the remainder of the algorithm and MUST include the value of the "d=" tag from each checked DKIM signature.
4. Perform SPF validation checks. The results of this step are passed to the remainder of the algorithm and MUST include the domain name used to complete the SPF check.
5. Conduct identifier alignment checks. With authentication checks and policy discovery performed, the Mail Receiver checks if Authenticated Identifiers fall into alignment as described in [Section 3](#). If one or more of the Authenticated Identifiers align with the [RFC5322.From](#) domain, the message is considered to pass the DMARC mechanism check. All other conditions (authentication failures, identifier mismatches) are considered to be DMARC mechanism check failures.
6. Apply policy. Emails that fail the DMARC mechanism check are disposed of in accordance with the discovered DMARC policy of the Domain Owner. See [Section 5.3](#) for details.

Heuristics applied in the absence of use by a Domain Owner of either SPF or DKIM (e.g., [[Best-Guess-SPF](#)]) SHOULD NOT be used, as it may be the case that the Domain Owner wishes a Message Receiver not to consider the results of that underlying authentication protocol at all.

Handling of messages for which SPF and/or DKIM evaluation encounters a DNS error is left to the discretion of the Mail Receiver. Further discussion is available in [Section 5.6.3](#).

[5.6.3](#). Policy Discovery

As stated above, the DMARC mechanism uses DNS TXT records to advertise policy. Policy discovery is accomplished via a method similar to the method used for SPF records. This method and the important differences between DMARC and SPF mechanisms are discussed below.

To balance the conflicting requirements of supporting wildcarding, allowing subdomain policy overrides, and limiting DNS query load, the

following DNS lookup scheme is employed:

1. Mail Receivers MUST query the DNS for a DMARC TXT record at the DNS domain matching the one found in the [RFC5322](#).From domain in the message. A possibly empty set of records is returned.
2. Records that do not start with a "v=" tag that identifies the current version of DMARC are discarded.
3. If the set is now empty, the Mail Receiver MUST query the DNS for a DMARC TXT record at the DNS domain matching the Organizational Domain in place of the [RFC5322](#).From domain in the message (if different). This record can contain policy to be asserted for subdomains of the Organizational Domain. A possibly empty set of records is returned.
4. Records that do not start with a "v=" tag that identifies the current version of DMARC are discarded.
5. If the remaining set contains multiple records or no records, policy discovery terminates and DMARC processing is not applied to this message.
6. If a retrieved policy record does not contain a valid "p" tag, or contains an "sp" tag that is not valid, then:
 1. if an "rua" tag is present and contains at least one syntactically valid reporting URI, the Mail Receiver SHOULD act as if a record containing a valid "v" tag and "p=none" was retrieved, and continue processing;
 2. otherwise, the Mail Receiver applies no DMARC processing to this message.

If the set produced by the mechanism above contains no DMARC policy record (i.e., any indication that there is no such record as opposed to a transient DNS error), Mail Receivers SHOULD NOT apply the DMARC mechanism to the message.

If the [RFC5322](#).From domain does not exist in the DNS, Mail Receivers SHOULD direct the receiving SMTP server to reject the message. The choice of mechanism for such rejection and the implications of those choices are discussed in [Section 9.3](#).

Handling of DNS errors when querying for the DMARC policy record is left to the discretion of the Mail Receiver. For example, to ensure minimal disruption of mail flow, transient errors could result in delivery of the message ("fail open"), or they could result in the

message being temporarily rejected (i.e., an SMTP 4yx reply) which invites the sending MTA to try again after the condition has possibly cleared, allowing a definite DMARC conclusion to be reached ("fail closed").

5.6.4. Message Sampling

If the "pct" tag is present in the policy record, the Mail Receiver MUST NOT enact the requested policy ("p" tag or "sp" tag) on more than the stated percent of the totality of affected messages. However, regardless of whether or not the "pct" tag is present, the Mail Receiver MUST include all relevant message data in any reports produced.

If email is subject to the DMARC policy of "quarantine", the Mail Receiver SHOULD quarantine the message. If the email is not subject to the "quarantine" policy (due to the "pct" tag), the Mail Receiver SHOULD apply local message classification as normal.

If email is subject to the DMARC policy of "reject", the Mail Receiver SHOULD reject the message (see [Section 9.3](#)). If the email is not subject to the "reject" policy (due to the "pct" tag), the Mail Receiver SHOULD treat the email as though the "quarantine" policy applies. This behavior allows senders to experiment with progressively stronger policies without relaxing existing policy.

Mail receivers implement "pct" via statistical mechanisms that achieve a close approximation to the requested percentage and provide a representative sample across a reporting period.

5.6.5. Store Results of DMARC Processing

The results of Mail Receiver-based DMARC processing should be stored for eventual presentation back to the Domain Owner in the form of aggregate feedback reports. [Section 5.1](#) and [Section 6.2](#) discuss aggregate feedback.

5.7. Policy Enforcement Considerations

Mail Receivers MAY choose to reject or quarantine email even if email passes the DMARC mechanism check. The DMARC mechanism does not inform Mail Receivers whether an email stream is "good". Mail Receivers are encouraged to maintain anti-abuse technologies to combat the possibility of DMARC-enabled criminal campaigns.

Mail Receivers MAY choose to accept email that fails the DMARC mechanism check even if the Domain Owner has published a "reject" policy. Mail Receivers need to make a best effort not to increase

the likelihood of accepting abusive mail if they choose not to comply with a Domain Owner's reject, against policy. At a minimum, addition of the Authentication-Results header field (see [[AUTH-RESULTS](#)]) is RECOMMENDED when delivery of failing mail is done. When this is done, the DNS domain name thus recorded MUST be encoded as an A-label.

Mail Receivers are only obligated to report reject or quarantine policy actions in aggregate feedback reports that are due to DMARC policy. They are not required to report reject or quarantine actions that are the result of local policy. If local policy information is exposed, abusers can gain insight into the effectiveness and delivery rates of spam campaigns.

Final disposition of a message is always a matter of local policy. An operator that wishes to favor DMARC policy over SPF policy, for example, will disregard the SPF policy since enacting an SPF-determined rejection prevents evaluation of DKIM; DKIM might otherwise pass, satisfying the DMARC evaluation. There is a trade-off to doing so, namely acceptance and processing of the entire message body in exchange for the enhanced protection DMARC provides.

DMARC-compliant Mail Receivers typically disregard any mail handling directive discovered as part of an authentication mechanism (e.g., ADSP, SPF) where a DMARC record is also discovered that specifies a policy other than "none". Deviating from this practice introduces inconsistency among DMARC operators in terms of handling of the message. However, such deviation is not proscribed.

To enable Domain Owners to receive DMARC feedback without impacting existing mail processing, discovered policies of "p=none" SHOULD NOT modify existing mail disposition processing.

Mail Receivers SHOULD also implement reporting instructions of DMARC in place of any extensions to SPF or DKIM that might enable such reporting.

6. DMARC Feedback

Providing Domain Owners with visibility into how Mail Receivers implement and enforce the DMARC mechanism in the form of feedback is critical to establishing and maintaining accurate authentication deployments. When Domain Owners can see what effect their policies and practices are having, they are better willing and able to use quarantine and reject policies.

6.1. Verifying External Destinations

It is possible to specify destinations for the different reports that are outside the authority of the Domain Owner making the request. This is enabled to allow domains that do not operate mail servers to request reports and have them go someplace that is able to receive and process them.

Without checks, this would allow a bad actor to publish a DMARC policy record that requests reports be sent to a victim address, and then send a large volume of mail that will fail both DKIM and SPF checks to a wide variety of destinations, which will in turn flood the victim with unwanted reports. Therefore, a verification mechanism is included.

When a Mail Receiver discovers a DMARC policy in the DNS, and the Organizational Domain at which that record was discovered is not identical to the Organizational Domain of the host part of the authority component of a [\[URI\]](#) specified in the "rua" or "ruf" tag, the following verification steps SHOULD be taken:

1. Extract the host portion of the authority component of the URI. Call this the "destination host", and refers to a Report Receiver.
2. Prepend the string "_report._dmarc".
3. Prepend the domain name from which the policy was retrieved, after conversion to an A-label if needed.
4. Query the DNS for a TXT record at the constructed name. If the result of this request is a temporary DNS error of some kind (e.g., a timeout), the Mail Receiver MAY elect to temporarily fail the delivery so the verification test can be repeated later.
5. For each record returned, parse the result as a series of "tag=value" pairs, i.e., the same overall format as the policy record (see [Section 5.4](#)). In particular, the "v=DMARC1" tag is mandatory and MUST appear first in the list. Discard any that do not pass this test.
6. If the result includes no TXT resource records that pass basic parsing, a positive determination of the external reporting relationship cannot be made; stop.
7. If at least one TXT resource record remains in the set after parsing, then the external reporting arrangement was authorized by the Report Receiver.

8. If a "rua" or "ruf" tag is thus discovered, replace the corresponding value extracted from the domain's DMARC policy record with the one found in this record. This permits the Report Receiver to override the report destination. However, to prevent loops or indirect abuse, the overriding URI MUST use the same destination host from the first step.

For example, if a DMARC policy query for "blue.example.com" contained "rua=mailto:reports@red.example.net", the host extracted from the latter ("red.example.net") does not match "blue.example.com", so this procedure is enacted. A TXT query for "blue.example.com._report._dmarc.red.example.net" is issued. If a single reply comes back containing a tag of "v=DMARC1", then the relationship between the two is confirmed. Moreover, red.example.net has the opportunity to override the report destination requested by "blue.example.com" if needed.

Where the above algorithm fails to confirm that the external reporting was authorized by the Report Receiver, the URI MUST be ignored by the Mail Receiver generating the report. Further, if the confirming record includes a URI whose host is again different than the domain publishing that override, the Mail Receiver generating the report MUST NOT generate a report to either the original or the override URI.

A Report Receiver publishes such a record in its DNS if it wishes to receive reports for other domains.

A Report Receiver that is willing to receive reports for any domain can use a wildcard DNS record. For example, a TXT resource record at "*._report._dmarc.example.com" containing at least "v=DMARC1" confirms that example.com is willing to receive DMARC reports for any domain.

If the Report Receiver is overcome by volume, it can simply remove the confirming DNS record. However, due to positive caching, the change could take as long as the time-to-live on the record to go into effect.

A Mail Receiver might decide not to enact this procedure if, for example, it relies on a local list of domains for which external reporting addresses are permitted.

6.2. Aggregate Reports

The DMARC aggregate feedback report is designed to provide Domain Owners with precise insight into:

- o authentication results
- o corrective action that needs to be taken by Domain Owners, and
- o the effect of Domain Owner DMARC policy on email streams processed by Mail Receivers.

Aggregate DMARC feedback provides visibility into real-world email streams that Domain Owners need to make informed decisions regarding the publication of DMARC policy. When Domain Owners know what legitimate mail they are sending, what the authentication results are on that mail, and what forged mail receivers are getting, they can make better decisions about the policies they need and the steps they need to take to enable those policies. When Domain Owners set policies appropriately and understand their effects, Mail Receivers can act on them confidently.

Visibility comes in the form of daily (or more frequent) Mail Receiver-originated feedback reports that contain aggregate data on message streams relevant to the Domain Owner. This information includes data about messages that passed DMARC authentication as well as those that did not.

The format for these reports is defined in [Appendix C](#).

The report SHOULD include the following data:

- o The DMARC policy discovered and applied, if any
- o The selected message disposition
- o The identifier evaluated by SPF and the SPF result, if any
- o The identifier evaluated by DKIM and the DKIM result, if any
- o For both DKIM and SPF, in indication of whether the identifier was in alignment
- o Data for each sender subdomain separately from mail from the sender's organizational domain, even if there is no explicit subdomain policy.
- o Sending and receiving domains
- o The policy requested by the Domain Owner and the policy actually applied (if different)

- o The number of successful authentications
- o The counts of messages based on all messages received even if their delivery is ultimately blocked by other filtering agents

Note that Domain Owners or their agents may change the published DMARC policy for a domain or subdomain at any time. From a Mail Receiver's perspective this will occur during a reporting period and may be noticed during that period, at the end of that period when reports are generated, or during a subsequent reporting period, all depending on the Mail Receiver's implementation. Under these conditions it is possible that a Mail Receiver could do any of the following:

- o generate a single aggregate report for such a reporting period that includes message dispositions based on the old policy, or a mix of the two policies, even though the report only contains a single "policy_published" element;
- o generate multiple reports for the same period, one for each published policy occurring during the reporting period;
- o generate a report whose end time occurs when the updated policy was detected, regardless of any requested report interval.

Such policy changes are expected to be infrequent for any given domain, whereas more stringent policy monitoring requirements on the Mail Receiver would produce a very large burden at Internet scale. Therefore it is the responsibility of report consumers and Domain Owners to be aware of this situation and allow for such mixed reports during the propagation of the new policy to Mail Receivers.

Aggregate reports are most useful when they all cover a common time period. By contrast, correlation of these reports from multiple generators when they cover incongruent time periods is difficult or impossible. Report generators SHOULD, wherever possible, adhere to hour boundaries for the reporting period they are using. For example, starting a per-day report at 00:00; starting per-hour reports at 00:00, 01:00, 02:00; et cetera. Report Generators using a 24-hour report period are strongly encouraged to begin that period at 00:00 UTC, regardless of local timezone or time of report production, in order to facilitate correlation.

6.2.1. Discovery

A Mail Receiver discovers reporting requests when it looks up a DMARC policy record that corresponds to a [RFC5322](#) From: domain on received mail. The presence of the "rua" tag specifies where to send

feedback.

For more on the considerations given to DMARC discovery, see [Section 5.6.3](#).

[6.2.2](#). Transport

Where the URI specified in an "rua" tag does not specify otherwise, a Mail Receiver generating a feedback report SHOULD apply a secure transport mechanism.

The Mail Receiver, after preparing a report, MUST evaluate the provided reporting URIs in the order given. Any reporting URI that includes a size limitation exceeded by the generated report (after compression and after any encoding required by the particular transport mechanism) MUST NOT be used. An attempt MUST be made to deliver an aggregate report to every remaining URI, up to the receiver's limits on supported URIs.

If transport is not possible because the services advertised by the published URIs are not able to accept reports (e.g., the URI refers to a service that is unreachable, or all provided URIs specify size limits exceeded by the generated record), the Mail Receiver SHOULD send a short report (see [Section 6.2.2.3](#)) indicating that a report is available but could not be sent. The Mail Receiver MAY cache that data and try again later, or MAY discard data that could not be sent.

[6.2.2.1](#). Email

The message generated by the Mail Receiver MUST be a [[MIME](#)] formatted [[MAIL](#)] message. The aggregate report itself MUST be included in one of the parts of the message. A human-readable portion MAY be included as a MIME part (such as a text/plain part).

The aggregate data MUST be an XML file that SHOULD be subjected to GZIP compression. Declining to apply compression can cause the report to be too large for a receiver to process (a commonly-observed receiver limit is ten megabytes); doing the compression increases the chances of acceptance of the report at some compute cost. The aggregate data SHOULD be present using the media type "application/gzip" if compressed (see [[GZIP](#)]), and "text/xml" otherwise. The filename is typically constructed using the following ABNF:


```
filename = receiver "!" policy-domain "!" begin-timestamp
          "!" end-timestamp [ "!" unique-id ] "." extension

unique-id = 1*(ALPHA | DIGIT)

receiver = domain
          ; imported from \[MAIL\]

policy-domain = domain

begin-timestamp = 1*DIGIT
                  ; seconds since 00:00:00 UTC January 1, 1970
                  ; indicating start of the time range contained
                  ; in the report

end-timestamp = 1*DIGIT
                ; seconds since 00:00:00 UTC January 1, 1970
                ; indicating end of the time range contained
                ; in the report

extension = "xml" / "xml.gz"
```

The extension MUST be "xml" for a plain XML file, or "xml.gz" for an XML file compressed using GZIP.

"unique-id" allows an optional unique ID generated by the Mail Receiver to distinguish among multiple reports generated simultaneously by different sources within the same Domain Owner.

For example, this is a possible filename for the gzip file of a report to the Domain Owner "example.com" from the Mail Receiver "mail.receiver.example".

```
mail.receiver.example!example.com!1013662812!1013749130.gz
```

No specific MIME message structure is required. It is presumed that the aggregate reporting address will be equipped to extract MIME parts with the prescribed media type and filename and ignore the rest.

Email streams carrying DMARC feedback data MUST conform to the DMARC mechanism, thereby resulting in an aligned "pass" (see [Section 3.1.4](#)). This practice minimizes the risk of report consumers processing fraudulent reports.

The [RFC5322](#).Subject field for individual report submissions SHOULD conform to the following ABNF:


```
dmARC-subject = %x52.65.70.6f.72.74 1*FWS      ; "Report"
               %x44.6f.6d.61.69.6e.3a 1*FWS    ; "Domain:"
               domain-name 1*FWS                ; from RFC6376
               %x53.75.62.6d.69.74.74.65.72.3a ; "Submitter:"
               1*FWS domain-name 1*FWS
               %x52.65.70.6f.72.74.2d.49.44.3a ; "Report-ID:"
               msg-id                             ; from RFC5322
```

The first domain-name indicates the DNS domain name about which the report was generated. The second domain-name indicates the DNS domain name representing the Mail Receiver generating the report. The purpose of the Report-ID: portion of the field is to enable the Domain Owner to identify and ignore duplicate reports that might be sent by a Mail Receiver.

For instance, this is a possible Subject field for a report to the Domain Owner "example.com" from the Mail Receiver "mail.receiver.example". It is line-wrapped as allowed by [[MAIL](#)].

```
Subject: Report Domain: example.com
      Submitter: mail.receiver.example
      Report-ID: <2002.02.15.1>
```

This transport mechanism potentially encounters a problem when feedback data size exceeds maximum allowable attachment sizes for either the generator or the consumer. See [Section 6.2.2.3](#) for further discussion.

[6.2.2.2.](#) Other Methods

The specification as written allows for the addition of other registered URI schemes to be supported in later versions.

[6.2.2.3.](#) Error Reports

When a Mail Receiver is unable to complete delivery of a report via any of the URIs listed by the Domain Owner, the Mail Receiver SHOULD generate an error message. An attempt MUST be made to send this report to all listed "mailto" URIs and it MAY also be sent to any or all other listed URIs.

The error report MUST be formatted per [[MIME](#)]. A text/plain part MUST be included that contains field-value pairs such as those found in Section 2 of [[DSN](#)]. The fields required, which may appear in any order, are:

Report-Date: A [\[MAIL\]](#)-formatted date expression indicating when the transport failure occurred.

Report-Domain: The domain-name about which the failed report was generated.

Report-ID: The Report-ID: that the report tried to use.

Report-Size: The size, in bytes, of the report that was unable to be sent. This MUST represent the number of bytes that the Mail Receiver attempted to send. Where more than one transport system was attempted, the sizes may be different; in such cases, separate error reports MUST be generated so that this value matches the actual attempt that was made.

Submitter: The domain-name representing the Mail Receiver that generated, but was unable to submit, the report.

Submitting-URI: The URI(s) to which the Mail Receiver tried, but failed, to submit the report.

An additional text/plain part MAY be included that gives a human-readable explanation of the above, and MAY also include a URI that can be used to seek assistance.

[6.3.](#) Failure Reports

Failure reports are normally generated and sent almost immediately after the Mail Receiver detects an authentication failure. Rather than waiting for an aggregate report, these reports are useful for quickly notifying the Domain Owners when there is an authentication failure. Whether the failure is due to an infrastructure problem or the message is inauthentic, failure reports also provide more information about the failed message than is available in an aggregate report.

These reports SHOULD include any URI(s) from the message that failed authentication. These reports SHOULD include as much of the message and message header as is reasonable to support the Domain Owner's investigation into what caused the message to fail authentication and track down the sender.

When a Domain Owner requests failure reports for the purpose of forensic analysis, and the Mail Receiver is willing to provide such reports, the Mail Receiver generates and sends a message using the format described in [\[AFRF\]](#). This document updates the AFRF format as described in [Section 6.3.1](#).

The format for failure reports is defined in [\[AFRF\]](#).

The destination(s) and nature of the reports are defined by the "ruf" and "fo" tags as defined in [Section 5.3](#).

Where multiple URIs are selected to receive failure reports the report generator MUST make an attempt to deliver to each of them.

An obvious consideration is the denial of service attack that can be perpetrated by an attacker who sends numerous messages purporting to be from the intended victim Domain Owner but which fail both SPF and DKIM; this would cause participating Mail Receivers to send failure reports to the Domain Owner or its delegate in potentially huge volumes. Accordingly, participating Mail Receivers are encouraged to aggregate these reports as much as is practical, using the Incidents field of the Abuse Reporting Format ([\[ARF\]](#)). Various aggregation techniques are possible, including:

- o only send a report to the first recipient of multi-recipient messages;
- o store reports for a period of time before sending them, allowing detection, collection, and reporting of like incidents;
- o apply rate limiting, such as a maximum number of reports per minute that will be generated (and the remainder discarded).

[6.3.1](#). Reporting Format Update

Operators implementing this specification also implement an augmented version of [\[AFRF\]](#) as follows:

1. A DMARC failure report includes the following ARF header fields, with the indicated normative requirement levels:
 - * Identity-Alignment (REQUIRED; defined below)
 - * Delivery-Result (OPTIONAL)
 - * DKIM-Domain, DKIM-Identity, DKIM-Selector (REQUIRED if the message was signed by DKIM)
 - * DKIM-Canonicalized-Header, DKIM-Canonicalized-Body (OPTIONAL if the message was signed by DKIM)
 - * SPF-DNS (REQUIRED)

2. The "Identity-Alignment" field is defined to contain a comma-separated list of authentication mechanism names that produced an aligned identity, or the keyword "none" if none did. ABNF:

```
id-align      = "Identity-Alignment:" [CFWS]
                ( "none" /
                  dmarc-method *( [CFWS] "," [CFWS] dmarc-method ) )
                [CFWS]

dmarc-method = ( "dkim" / "spf" )
                ; each may appear at most once in an id-align
```

3. Authentication Failure Type "dmarc" is defined, which is to be used when a failure report is generated because some or all of the authentication mechanisms failed to produce aligned identifiers. Note that a failure report generator MAY also independently produce an AFRF message for any or all of the underlying authentication methods.

7. Minimum Implementations

A minimum implementation of DMARC has the following characteristics:

- o Is able to send and/or receive reports at least daily;
- o Is able to send and/or receive reports using "mailto" URIs;
- o Other than in exceptional circumstances such as resource exhaustion, can generate or accept a report up to ten megabytes in size;
- o If acting as a Mail Receiver, fully implements the provisions of [Section 5.6](#).

8. Privacy Considerations

This section discusses security issues specific to private data that may be included in the interactions that are part of DMARC.

8.1. Data Exposure Considerations

Aggregate reports are limited in scope to DMARC policy and disposition results, to information pertaining to the underlying authentication mechanisms, and to the identifiers involved in DMARC validation.

Failed message reporting provides message-specific details pertaining to authentication failures. Individual reports can contain message content as well as trace header fields. Domain Owners are able to analyze individual reports and attempt to determine root causes of authentication mechanism failures, gain insight into misconfigurations or other problems with email and network infrastructure, or inspect messages for insight into abusive practices.

Both report types may expose sender and recipient identifiers (e.g., [RFC5322](#).From addresses), and although the [\[AFRE\]](#) format used for failed message reporting supports redaction, failed message reporting is capable of exposing the entire message to the report recipient.

Domain Owners requesting reports will receive information about mail claiming to be from them, which includes mail that was not, in fact, from them. Information about the final destination of mail where it might otherwise be obscured by intermediate systems will therefore be exposed.

Domain Owners requesting reports will also receive information about mail forwarded to unexpected Mail Receivers when message forwarding relationships exists.

Disclosure of information about the messages is being requested by the entity generating the email in the first place, i.e., the Domain Owner and not the Mail Receiver, so this may not fit squarely within existing privacy policy provisions. For some providers, aggregate and failed message reporting are viewed as a function similar to complaint reporting about spamming or phishing, and treated similarly under the privacy policy. Report generators (i.e., Mail Receivers) are encouraged to review their reporting limitations under such policies before enabling DMARC reporting.

[8.2.](#) Report Recipients

A DMARC record can specify that reports should be sent to an intermediary operating on behalf of the Domain Owner. This is done when the Domain Owner contracts with an entity to monitor mail-streams for abuse and performance issues. Receipt by third parties of such data may or may not be permitted by the Mail Receiver's privacy policy, terms of use, or other similar governing document. Domain Owners and Mail Receivers should both review and understand if their own internal policies constrain the use and transmission of DMARC reporting.

8.3. Secure Protocols

This document encourages use of secure transport mechanisms to prevent loss of private data to third parties that may be able to monitor such transmissions. Unencrypted mechanisms should be avoided.

In particular, a message that was originally encrypted or otherwise secured might appear in a report that is not sent securely, which could reveal private information.

9. Other Topics

This section discusses some topics regarding choices made in the development of DMARC, largely to commit the history to record.

9.1. Issues Specific to SPF

Though DMARC does not inherently change the semantics of an SPF policy record, historically lax enforcement of such policies has led many to publish extremely broad records containing many large network ranges. Domain Owners are strongly encouraged to carefully review their SPF records to understand which networks are authorized to send on behalf of the Domain Owner before publishing a DMARC record.

Some receiver architectures might implement SPF in advance of any DMARC operations. This means a "-" prefix on a Sender's SPF mechanism, such as "-all", could cause that rejection go into effect early in handling, causing message rejection, before any DMARC processing takes place. Operators choosing to use "-all" should be aware of this.

9.2. DNS Load and Caching

DMARC policies are communicated using the DNS, and therefore inherit a number of considerations related to DNS caching. The inherent conflict between freshness and the impact of caching on the reduction of DNS-lookup overhead should be considered from the Mail Receiver's point of view. Should Domain Owners publish a DNS record with a very short TTL, Mail Receivers can be provoked through the injection of large volumes of messages to overwhelm the Domain Owner's DNS. Although this is not a concern specific to DMARC, the implications of a very short TTL should be considered when publishing DMARC policies.

Conversely, long TTLs will cause records to be cached for long periods of time. This can cause a critical change to DMARC parameters advertised by a Domain Owner to go unnoticed for the

length of the TTL (while waiting for DNS caches to expire). Avoiding this problem can mean shorter TTLs, with the potential problems described above. A balance should be sought to maintain responsiveness of DMARC preference changes while preserving the benefits of DNS caching.

9.3. Rejecting Messages

This proposal calls for rejection of a message during the SMTP session under certain circumstances. This is preferable to generation of a Delivery Status notification ([\[DSN\]](#)) since fraudulent messages caught and rejected using DMARC would then result in annoying generation of such failure reports that go back to the [RFC5321](#).MailFrom address.

This synchronous rejection is typically done in one of two ways:

- o Full rejection, wherein the SMTP server issues a 5xy reply code as an indication to the SMTP client that the transaction failed; the SMTP client is then responsible for generating notification that delivery failed (see Section 4.2.5 of [\[SMTP\]](#)).
- o A "silent discard", wherein the SMTP server returns a 2xy reply code implying to the client that delivery (or, at least, relay) was successfully completed, but then simply discarding the message with no further action.

Each of these has a cost. For instance, a silent discard can help to prevent backscatter, but it also effectively means the SMTP server has to be programmed to give a false result, which can confound external debugging efforts.

Similarly, the text portion of the SMTP reply may be important to consider. For example, when rejecting a message, revealing the reason for the rejection might give an attacker enough information to bypass those efforts on a later attempt, though it might also assist a legitimate client to determine the source of some local issue that caused the rejection.

In the latter case, when doing an SMTP rejection, providing a clear hint can be useful in resolving issues. A receiver might indicate in plain text the reason for the rejection by using the word "DMARC" somewhere in the reply text. Many systems are able to scan the SMTP reply text to determine the nature of the rejection, thus providing a machine-detectable reason for rejection allows automated sorting of rejection causes so they can be properly addressed. For example:

550 5.7.1 Email rejected per DMARC policy for example.com

If a Mail Receiver elects to defer delivery due to inability to retrieve or apply DMARC policy, this is best done with a 4xy SMTP reply code.

9.4. Identifier Alignment Considerations

The DMARC mechanism allows both DKIM and SPF-authenticated identifiers to authenticate email on behalf of a Domain Owner and, possibly, on behalf of different subdomains. If malicious or unaware users can gain control of the SPF record or DKIM signing practices for a subdomain, the subdomain can be used to generate DMARC-passing email on behalf of the Organizational Domain.

For example, an attacker who controls the SPF record for "evil.example.com" can send mail with an [RFC5322](#).From field containing "foo@example.com" that can pass both authentication and the DMARC check against "example.com".

The Organizational Domain administrator should be careful not to delegate control of sub-domains if this is an issue, and to consider using the "strict" Identifier Alignment option if appropriate.

10. IANA Considerations

This section describes actions requested of IANA.

10.1. Authentication-Results Method Registry Update

IANA is requested to add the following to the Email Authentication Method Name Registry:

Method: dmarc

Defined In: [this document]

ptype: header

property: from

value: the domain portion of the [RFC5322](#).From field

10.2. Authentication-Results Result Registry Update

IANA has added the following in the Email Authentication Result Name Registry:

Code: none

Existing/New Code: existing

Defined In: [[AUTH-RESULTS](#)]

Auth Method: dmarc (added)

Meaning: No DMARC policy record was published for the aligned identifier, or no aligned identifier could be extracted.

Code: pass

Existing/New Code: existing

Defined In: [[AUTH-RESULTS](#)]

Auth Method: dmarc (added)

Meaning: A DMARC policy record was published for the aligned identifier, and at least one of the authentication mechanisms passed.

Code: fail

Existing/New Code: existing

Defined In: [[AUTH-RESULTS](#)]

Auth Method: dmarc (added)

Meaning: A DMARC policy record was published for the aligned identifier, and none of the authentication mechanisms passed.

Code: temperror

Existing/New Code: existing

Defined In: [[AUTH-RESULTS](#)]

Auth Method: dmarc (added)

Meaning: A temporary error occurred during DMARC evaluation. A later attempt might produce a final result.

Code: permerror

Existing/New Code: existing

Defined In: [[AUTH-RESULTS](#)]

Auth Method: dmarc (added)

Meaning: A permanent error occurred during DMARC evaluation, such as encountering a syntactically incorrect DMARC record. A later attempt is unlikely to produce a final result.

10.3. Feedback Report Header Fields Registry Update

The following is added to the Feedback Report Header Fields Registry:

Field Name: Identity-Alignment

Description: indicates whether the message about which a report is being generated had any identifiers in alignment as defined in [this RFC]

Multiple Appearances: no

Related "Feedback-Type": auth-failure

Published In: [this RFC]

Status: current

10.4. DMARC Tag Registry

Names of DMARC tags must be registered with IANA. New entries are assigned only for values that have been documented in a manner that satisfies the terms of Specification Required, per [[IANA-CONSIDERATIONS](#)]. Each registration must include the tag name, the specification that defines it, a brief description, and its status which must be one of "current", "experimental" or "historic". The Designated Expert needs to confirm that the provided specification adequately describes the new tag and clearly presents how it would be used within the DMARC context by Domain Owners and Mail Receivers.

To avoid version compatibility issues, tags added to the DMARC specification are to avoid changing the semantics of existing records when processed by implementations conforming to prior specifications.

The initial set of entries in this registry is as follows:

Tag Name	Defined	Status	Description
adkim	[THIS MEMO]	current	DKIM alignment mode
aspf	[THIS MEMO]	current	SPF alignment mode
fo	[THIS MEMO]	current	Failure reporting options
pct	[THIS MEMO]	current	Sampling rate
p	[THIS MEMO]	current	Requested handling policy
rf	[THIS MEMO]	current	Failure reporting format(s)
ri	[THIS MEMO]	current	Aggregate Reporting interval
rua	[THIS MEMO]	current	Reporting URI(s) for aggregate data
ruf	[THIS MEMO]	current	Reporting URI(s) for failure data
sp	[THIS MEMO]	current	Requested handling policy for subdomains
v	[THIS MEMO]	current	Specification version

10.5. DMARC Report Format Registry

Names of DMARC failure reporting formats must be registered with IANA. New entries are assigned only for values that satisfy the definition of Specification Required, per [\[IANA-CONSIDERATIONS\]](#). In addition to a reference to a permanent specification, each registration must include the tag name, the specification that defines it, a brief description, and its status which must be one of "current", "experimental" or "historic". The Designated Expert needs to confirm that the provided specification adequately describes the report format and clearly presents how it would be used within the DMARC context by Domain Owners and Mail Receivers.

The initial set of entries in this registry is as follows:

Format	Defined	Status	Description
Name			
afrf	[THIS MEMO]	current	Authentication Failure Reporting Format (see [AFRF])

11. Security Considerations

This section discusses security issues and possible remediations (where available) for DMARC.

11.1. Authentication Methods

Security considerations from the authentication methods used by DMARC are incorporated here by reference.

11.2. Attacks on Reporting URIs

URIs published in DNS TXT records are well-understood possible targets for attack. Specifications such as [DNS] and [ROLES] either expose or cause the exposure of email addresses that could be flooded by an attacker, for example; MX, NS and other records found in the DNS advertise potential attack destinations; common DNS names such as "www" plainly identify the locations at which particular services can be found, providing destinations for targeted denial-of-service or penetration attacks.

Thus, Domain Owners will need to harden these addresses against various attacks, including but not limited to:

- o high-volume denial-of-service attacks;
- o deliberate construction of malformed reports intended to identify or exploit parsing or processing vulnerabilities;
- o deliberate construction of reports containing false claims for the Submitter or Reported-Domain fields, including the possibility of false data from compromised but known Mail Receivers.

11.3. DNS Security

The DMARC mechanism and its underlying technologies (SPF, DKIM) depend on the security of the DNS. To reduce the risk of subversion of the DMARC mechanism due to DNS-based exploits, serious

consideration should be given to the deployment of DNSSEC in parallel to the deployment of DMARC by both Domain Owners and Mail Receivers.

11.4. Display Name Attacks

A common attack in messaging abuse is the presentation of false information in the display-name portion of the [RFC5322](#).From field. For example, it is possible for the email address in that field to be an arbitrary address or domain name, while containing a well-known name (a person, brand, role, etc.) in the display name, intending to fool the end user into believing that the name is used legitimately. The attack is predicated on the notion that most common MUAs will show the display name and not the email address when both are available.

Generally, display name attacks are out of scope for DMARC as further exploration of possible defenses against these attacks needs to be undertaken.

There are a few possible mechanisms that attempt mitigation of these attacks, such as:

- o If the display name is found to include an email address (as specified in [[MAIL](#)]), execute the DMARC mechanism on the domain name found there rather than the domain name discovered originally. However, this addresses only a very specific attack space and is easily circumvented by spoofers simply by not using an email address in the display name. There are also known cases of legitimate uses of an email address in the display name with a domain different from the one in the address portion, e.g.:

From: "user@example.org via Bug Tracker" <support@example.com>

- o In the MUA, only show the display name if the DMARC mechanism succeeds. This too is easily defeated, as an attacker could arrange to pass the DMARC tests while fraudulently using another domain name in the display name.
- o In the MUA, only show the display name if the DMARC mechanism passes and the email address thus validated matches one found in the receiving user's list of known addresses.

11.5. External Reporting Addresses

To avoid abuse by bad actors, reporting addresses generally have to be inside the domains about which reports are requested. In order to accommodate special cases such as a need to get reports about domains that cannot actually receive mail, [Section 6.1](#) describes a DNS-based

mechanism for verifying approved external reporting.

The obvious consideration here is an increased DNS load against domains that are claimed as external recipients. Negative caching will mitigate this problem, but only to a limited extent, mostly dependent on the default time-to-live in the domain's SOA record.

Where possible, external reporting is best achieved by having the report be directed to domains that can receive mail and simply having it automatically forwarded to the desired external destination.

Note that the addresses shown in the "ruf" tag receive more information that might be considered private data, since it is possible for actual email content to appear in the failure reports. The URIs identified there are thus more attractive targets for intrusion attempts than those found in the "rua" tag. Moreover, attacking the DNS of the subject domain to cause failure data to be routed fraudulently to an attacker's systems may be an attractive prospect. Deployment of [\[DNSSEC\]](#) is advisable if this is a concern.

The verification mechanism presented in [Section 6.1](#) is currently not mandatory ("MUST") but strongly recommended ("SHOULD"). It is possible that it would be elevated to a "MUST" by later security review.

[12.](#) References

[12.1.](#) Normative References

- [ABNF] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", [RFC 5234](#), January 2008.
- [AFRF] Fontana, H., "Authentication Failure Reporting using the Abuse Report Format", [RFC 6591](#), April 2012.
- [AFRF-DKIM] Kucherawy, M., "Extensions to DomainKeys Identified Mail (DKIM) for Failure Reporting", [RFC 6651](#), June 2012.
- [AFRF-SPF] Kitterman, S., "Sender Policy Framework (SPF) Authentication Failure Reporting Using the Abuse Reporting Format", [RFC 6652](#), June 2012.
- [DKIM] Crocker, D., Hansen, T., and M. Kucherawy, "DomainKeys Identified Mail (DKIM) Signatures", [RFC 6376](#), September 2011.

- [DNS] Mockapetris, P., "Domain names - implementation and specification", STD 13, [RFC 1035](#), November 1987.
- [DNS-CASE] Eastlake, D., "Domain Name System (DNS) Case Insensitivity Clarification", [RFC 4343](#), January 2006.
- [GZIP] Levine, J., "The 'application/zlib' and 'application/gzip' Media Types", [RFC 6713](#), August 2012.
- [IDNA] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", [RFC 5890](#), August 2000.
- [KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [MAIL] Resnick, P., Ed., "Internet Message Format", [RFC 5322](#), October 2008.
- [MIME] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", [RFC 2045](#), November 1996.
- [SEC-TERMS] Shirey, R., "Internet Security Glossary, Version 2", [RFC 4949](#), August 2007.
- [SMTP] Klensin, J., "Simple Mail Transfer Protocol", [RFC 5321](#), October 2008.
- [SPF] Wong, M. and W. Schlitt, "Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1", [RFC 4408](#), April 2006.
- [URI] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", [RFC 3986](#), January 2005.

[12.2.](#) Informative References

- [ADSP] Allman, E., Fenton, J., Delany, M., and J. Levine, "DomainKeys Identified Mail (DKIM) Author Domain Signing Practices (ADSP)", [RFC 5617](#), August 2009.
- [ARF] Shafranovich, Y., Levine, J., and M. Kucherawy, "An Extensible Format for Email Feedback Reports", [RFC 5965](#),

August 2010.

[AUTH-RESULTS]

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

[Best-Guess-SPF]

Kitterman, S., "Sender Policy Framework: Best guess record (FAQ entry)", May 2010,
<http://www.openspf.org/FAQ/Best_guess_record>.

[DKIM-DEPLOYMENT]

Hansen, T., Siegel, E., Crocker, D., and P. Hallam-Baker, "DomainKeys Identified Mail (DKIM) Development, Deployment, and Operations", [RFC 5863](#), May 2010.

[DKIM-OVERVIEW]

Hansen, T., Crocker, D., and P. Hallam-Baker, "DomainKeys Identified Mail (DKIM) Service Overview", [RFC 5585](#), July 2009.

[DKIM-THREATS]

Fenton, J., "Analysis of Threats Motivating DomainKeys Identified Mail (DKIM)", [RFC 4686](#), September 2006.

[DNSSEC] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", [RFC 4033](#), March 2005.

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

[EMAIL-ARCH]

Crocker, D., "Internet Mail Architecture", [RFC 5598](#), July 2009.

[IANA-CONSIDERATIONS]

Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 5226](#), May 2008.

[ROLES] Crocker, D., "Mailbox Names for Common Services, Roles and Functions", [RFC 2142](#), May 1997.

URIs

[1] <<http://dmarc.org>>

[Appendix A](#). Technology Considerations

This section documents some design decisions that were made in the development of DMARC. Specifically, addressed here are some suggestions that were considered but not included in the design. This text is included to explain why they were considered and not included in this version.

[A.1](#). S/MIME

S/MIME, or Secure Multipurpose Internet Mail Extensions, is a standard for encryption and signing of MIME data in a message. This was suggested and considered as a third security protocol for authenticating the source of a message.

DMARC is focused on authentication at the domain level (i.e., the Domain Owner taking responsibility for the message), while S/MIME is really intended for user-to-user authentication and encryption. This alone appears to make it a bad fit for DMARC's goals.

S/MIME also suffers from the heavyweight problem of Public Key Infrastructure, which means distribution of keys used to verify signatures needs to be incorporated. In many instances, this alone is a showstopper. There have been consistent promises that PKI usability and deployment will improve, but these have yet to materialize. DMARC can revisit this choice after those barriers are addressed.

S/MIME has extensive deployment in specific market segments (government, for example), but does not enjoy similar widespread deployment over the general Internet, and this shows no signs of changing. DKIM and SPF both are deployed widely over the general Internet and their adoption rates continue to be positive.

Finally, experiments have shown that including S/MIME support in the initial version of DMARC would neither cause nor enable a substantial increase in the accuracy of the overall mechanism.

[A.2](#). Method Exclusion

It was suggested that DMARC include a mechanism by which a Domain Owner could tell Message Receivers not to attempt validation by one of the supported methods (e.g., "check DKIM, but not SPF").

Specifically, consider a Domain Owner that has deployed one of the technologies, and that technology fails for some messages, but such failures don't cause enforcement action. Deploying DMARC would cause enforcement action for policies other than "none", which would appear

to exclude participation by that Domain Owner.

The DMARC development team evaluated the idea of policy exception mechanisms on several occasions and invariably concluded that there was not a strong enough use case to include them. The specific target audience for DMARC does not appear to have concerns about the failure modes of one or the other being a barrier to DMARC's adoption.

In the scenario described above, the Domain Owner has a few options:

1. Tighten up its infrastructure to minimize the failure modes of the single deployed technology.
2. Deploy the other supported authentication mechanism, to offset the failure modes of the first.
3. Deploy DMARC in a reporting-only mode.

A.3. Sender Header Field

It has been suggested in several message authentication efforts that the Sender header field be checked for an identifier of interest, as the standards indicate this as the proper way to indicate a re-mailing of content such as through a mailing list. Most recently, it was a protocol-level option for DomainKeys, but on evolution to DKIM, this property was removed.

The DMARC development team considered this and decided not to include support for doing so, for the following reasons:

1. The main user protection approach is to be concerned with what the user sees when a message is rendered. There is no consistent behavior among MUAs regarding what to do with the content of the Sender field, if present. Accordingly, supporting checking of the Sender identifier would mean applying policy to an identifier the end user might never actually see, which can create a vector for attack against end users by simply forging a Sender field containing some identifier that DMARC will like.
2. Although it is certainly true that this is what Sender is for, its use in this way is also unreliable, making it a poor candidate for inclusion in the DMARC evaluation algorithm.
3. Allowing multiple ways to discover policy introduces unacceptable ambiguity into the DMARC evaluation algorithm in terms of which policy is to be applied and when.

A.4. Domain Existence Test

A common practice among MTA operators, and indeed one documented in [ADSP], is a test to determine domain existence prior to any more expensive processing. This is typically done by querying the DNS for MX, A or AAAA resource records for the name being evaluated, and assuming the domain is non-existent if it could be determined that no such records were published for that domain name.

The original pre-standardization version of this protocol included a mandatory check of this nature. It was ultimately removed, as the method's error rate was too high without substantial manual tuning and heuristic work. There are indeed use cases this work needs to address where such a method would return a negative result about a domain for which reporting is desired, such as a registered domain name that never sends legitimate mail and thus has none of these records present in the DNS.

A.5. Issues With ADSP In Operation

DMARC has been characterized as a "super-ADSP" of sorts.

Contributors to DMARC have compiled a list of issues associated with ADSP, gained from operational experience, that have influenced the direction of DMARC:

1. ADSP has no support for subdomains, i.e., the ADSP record for example.com does not explicitly or implicitly apply to subdomain.example.com. If wilddcarding is not applied, then spammers can trivially bypass ADSP by sending from a subdomain with no ADSP record.
2. Non-existent subdomains are explicitly out of scope in ADSP. There is nothing in ADSP that states receivers should simply reject mail from NXDOMAINs regardless of ADSP policy (which of course allows spammers to trivially bypass ADSP by sending email from non-existent subdomains).
3. ADSP has no operational advice on when to look up the ADSP record.
4. ADSP has no support for using SPF as an auxiliary mechanism to DKIM.
5. ADSP has no support for a slow roll-out, i.e., no way to configure a percentage of email on which the receiver should apply the policy. This is important for large-volume senders.

6. ADSP has no explicit support for an intermediate phase where the receiver quarantines (e.g., sends to the recipient's "spam" folder) rather than rejects the email.
7. The binding between the "From" header domain and DKIM is too tight for ADSP; they must match exactly.

A.6. Organizational Domain Discovery Issues

Although protocols like ADSP are useful for "protecting" a specific domain name, they are not helpful at protecting subdomains. If one wished to protect "example.com" by requiring via ADSP that all mail bearing an [RFC5322](#).From domain of "example.com" be signed, this would "protect" that domain; however, one could then craft an email whose [RFC5322](#).From domain is "security.example.com", and ADSP would not provide any protection. One could use a DNS wildcard, but this can undesirably interfere with other DNS activity; one could add ADSP records as fraudulent domains are discovered, but this solution does not scale and is a purely reactive measure against abuse.

The DNS does not provide a method by which the "domain of record", or the domain that was actually registered with a domain registrar, can be determined given an arbitrary domain name. Suggestions have been made that attempt to glean such information from SOA or NS resource records, but these too are not fully reliable as the partitioning of the DNS is not always done at administrative boundaries.

When seeking domain-specific policy based on an arbitrary domain name, one could "climb the tree", dropping labels off the left end of the name until the root is reached or a policy is discovered, but then one could craft a name that has a large number of nonsense labels; this would cause a Mail Receiver to attempt a large number of queries in search of a policy record. Sending many such messages constitutes an amplified denial-of-service attack.

The Organizational Domain mechanism is a necessary component to the goals of DMARC. The method described in [Section 3](#) is far from perfect, but serves this purpose reasonably well without adding undue burden or semantics to the DNS. If a method is created to do so that is more reliable and secure than the use of a public suffix list, DMARC should be amended to use that method as soon as it is generally available.

A.6.1. Public Suffix Lists

A public suffix list for the purposes of determining the Organizational Domain can be obtained from various sources. The most common one is maintained by the Mozilla Foundation and made public at

<http://publicsuffix.org>. License terms governing the use of that list are available at that URI.

Note that if operators use a variety of public suffix lists, interoperability will be difficult or impossible to guarantee.

[Appendix B](#). Examples

This section illustrates both the Domain Owner side and the Mail Receiver side of a DMARC exchange.

[B.1](#). Identifier Alignment examples

The following examples illustrate the DMARC mechanism's use of Identifier Alignment. For brevity's sake, only message headers are shown as message bodies are not considered when conducting DMARC checks.

[B.1.1](#). SPF

The following SPF examples assume that SPF produces a passing result.

Example 1: SPF in alignment:

```
MAIL FROM: <sender@example.com>

From: sender@example.com
Date: Fri, Feb 15 2002 16:54:30 -0800
To: receiver@example.org
Subject: here's a sample
SPF In Alignment
```

In this case, the [RFC5321](#).MailFrom parameter and the [RFC5322](#).From field have identical DNS domains. Thus, the identifiers are in alignment.

Example 2: SPF in alignment (parent):

```
MAIL FROM: <sender@child.example.com>

From: sender@example.com
Date: Fri, Feb 15 2002 16:54:30 -0800
To: receiver@example.org
Subject: here's a sample
SPF In Alignment (Parent)
```

In this case, the [RFC5322](#).From parameter includes a DNS domain that

is a parent of the [RFC5321](#).MailFrom domain. Thus, the identifiers are in alignment if "relaxed" SPF mode is requested by the Domain Owner, and not in alignment if "strict" SPF mode is requested.

Example 3: SPF not in alignment:

```
MAIL FROM: <sender@example.net>

From: sender@child.example.com
Date: Fri, Feb 15 2002 16:54:30 -0800
To: receiver@example.org
Subject: here's a sample
SPF Not In Alignment
```

In this case, the [RFC5321](#).MailFrom parameter includes a DNS domain that is neither the same as nor a parent of the [RFC5322](#).From domain. Thus, the identifiers are not in alignment.

[B.1.2](#). DKIM

The examples below assume the DKIM signatures pass verification. Alignment cannot exist with a DKIM signature that does not verify.

Example 1: DKIM in alignment:

```
DKIM-Signature: v=1; ...; d=example.com; ...
From: sender@example.com
Date: Fri, Feb 15 2002 16:54:30 -0800
To: receiver@example.org
Subject: here's a sample
DKIM In Alignment
```

In this case, the DKIM "d=" parameter and the [RFC5322](#).From field have identical DNS domains. Thus, the identifiers are in alignment.

Example 2: DKIM in alignment (parent):

```
DKIM-Signature: v=1; ...; d=example.com; ...
From: sender@child.example.com
Date: Fri, Feb 15 2002 16:54:30 -0800
To: receiver@example.org
Subject: here's a sample
DKIM In Alignment (Parent)
```

In this case, the DKIM signature's "d=" parameter includes a DNS domain that is a parent of the [RFC5322](#).From domain. Thus, the identifiers are in alignment for "relaxed" mode, but not for "strict" mode.

Example 3: DKIM not in alignment:

```
DKIM-Signature: v=1; ...; d=sample.net; ...
From: sender@child.example.com
Date: Fri, Feb 15 2002 16:54:30 -0800
To: receiver@example.org
Subject: here's a sample
DKIM Not In Alignment
```

In this case, the DKIM signature's "d=" parameter includes a DNS domain that is neither the same as nor a parent of the [RFC5322](#).From domain. Thus, the identifiers are not in alignment.

[B.2.](#) Domain Owner example

A Domain Owner that wants to use DMARC should have already deployed and tested SPF and DKIM. The next step is to publish a DNS record that advertises a DMARC policy for the Domain Owner's organizational domain.

[B.2.1.](#) Entire Domain, Monitoring Only

The owner of the domain "example.com" has deployed SPF and DKIM on its messaging infrastructure. The owner wishes to begin using DMARC with a policy that will solicit aggregate feedback from receivers without affecting how the messages are processed, in order to:

- o Confirm that its legitimate messages are authenticating correctly
- o Verify that all authorized message sources have implemented authentication measures
- o Determine how many messages from other sources would be affected by a blocking policy

The Domain Owner accomplishes this by constructing a policy record indicating that:

- o The version of DMARC being used is "DMARC1" ("v=DMARC1")
- o Receivers should not alter how they treat these messages because of this DMARC policy record ("p=none")
- o Aggregate feedback reports should be sent via email to the address "dmarc-feedback@example.com" ("rua=mailto:dmarc-feedback@example.com")

- o All messages from this organizational domain are subject to this policy (no "pct" tag present, so the default of 100% applies)

The DMARC policy record might look like this when retrieved using a common command-line tool:

```
% dig +short TXT _dmarc.example.com.  
"v=DMARC1; p=none; rua=mailto:dmarc-feedback@example.com"
```

To publish such a record, the DNS administrator for the Domain Owner creates an entry like the following in the appropriate zone file (following the conventional zone file format):

```
; DMARC record for the domain example.com  
  
_dmarc IN TXT ( "v=DMARC1; p=none; "  
                "rua=mailto:dmarc-feedback@example.com" )
```

B.2.2. Entire Domain, Monitoring Only, Per-Message Reports

The Domain Owner from the previous example has used the aggregate reporting to discover some messaging systems that had not yet implemented DKIM correctly, but they are still seeing periodic authentication failures. In order to diagnose these intermittent problems they wish to request per-message failure reports when authentication failures occur.

Not all Receivers will honor such a request, but the Domain Owner feels that any reports it does receive will be helpful enough to justify publishing this record. The default per-message report format ([\[AFRE\]](#)) meets the Domain Owner's needs in this scenario.

The Domain Owner accomplishes this by adding the following to its policy record from [Appendix B.2](#)):

- o Per-message failure reports should be sent via email to the address "auth-reports@example.com"
("ruf=mailto:auth-reports@example.com")

The DMARC policy record might look like this when retrieved using a common command-line tool (the output shown would appear on a single line, but is wrapped here for publication):

```
% dig +short TXT _dmarc.example.com.  
"v=DMARC1; p=none; rua=mailto:dmarc-feedback@example.com;  
ruf=mailto:auth-reports@example.com"
```

To publish such a record, the DNS administrator for the Domain Owner

might create an entry like the following in the appropriate zone file (following the conventional zone file format):

```
; DMARC record for the domain example.com

_dmarc IN TXT ( "v=DMARC1; p=none; "
                "rua=mailto:dmarc-feedback@example.com; "
                "ruf=mailto:auth-reports@example.com" )
```

[B.2.3.](#) Per-Message Failure Reports Directed to Third Party

The Domain Owner from the previous example is maintaining the same policy, but now wishes to have a third party receive and process the per-message failure reports. Again, not all Receivers will honor this request, but those that do may implement additional checks to validate that the third party wishes to receive the failure reports for this domain.

The Domain Owner needs to alter its policy record from [Appendix B.2.2](#) as follows:

- o Per message failure reports should be send via email to the address "auth-reports@thirdparty.example.net" ("ruf=mailto:auth-reports@thirdparty.example.net")

The DMARC policy record might look like this when retrieved using a common command-line tool (the output shown would appear on a single line, but is wrapped here for publication):

```
% dig +short TXT _dmarc.example.com.
"v=DMARC1; p=none; rua=mailto:dmarc-feedback@example.com;
ruf=mailto:auth-reports@thirdparty.example.net"
```

To publish such a record, the DNS administrator for the Domain Owner might create an entry like the following in the appropriate zone file (following the conventional zone file format):

```
; DMARC record for the domain example.com

_dmarc IN TXT ( "v=DMARC1; p=none; "
                "rua=mailto:dmarc-feedback@example.com; "
                "ruf=mailto:auth-reports@thirdparty.example.net" )
```

Because the address used in the "ruf" tag is outside the Organizational Domain in which this record is published, conforming Receivers will implement additional checks as described in [Section 6.1](#) of this document. In order to pass these additional checks, the third party will need to publish an additional DNS record

as follows:

- o Given the DMARC record published by the Domain Owner at "_dmarc.example.com", the DNS administrator for the third party will need to publish a TXT resource record at "example.com._report._dmarc.thirdparty.example.net" with the value "v=DMARC1".

The resulting DNS record might look like this when retrieved using a common command-line tool (the output shown would appear on a single line, but is wrapped here for publication):

```
% dig +short TXT example.com._report._dmarc.thirdparty.example.net
"v=DMARC1"
```

To publish such a record, the DNS administrator for example.net might create an entry like the following in the appropriate zone file (following the conventional zone file format):

```
; zone file for thirdparty.example.net
; Accept DMARC failure reports on behalf of example.com
```

```
example.com._report._dmarc    IN    TXT    "v=DMARC1"
```

Intermediaries and other third parties should refer to [Section 6.1](#) for the full details of this mechanism.

B.2.4. Sub-Domain, Sampling, and Multiple Aggregate Report URIs

The Domain Owner has implemented SPF and DKIM in a sub-domain used for pre-production testing of messaging services. It now wishes to request that participating receivers act to reject messages from this sub-domain that fail to authenticate.

As a first step it will ask that a portion (1/4 in this example) of failing messages be quarantined, enabling examination of messages sent to mailboxes hosted by participating receivers. Aggregate feedback reports will be sent to a mailbox within the Organizational Domain, and to a mailbox at a third party selected and authorized to receive same by the Domain Owner. Aggregate reports sent to the third party are limited to a maximum size of ten megabytes.

The Domain Owner will accomplish this by constructing a policy record indicating that:

- o The version of DMARC being used is "DMARC1" ("v=DMARC1")

- o It is applied only to this sub-domain (record is published at "_dmarc.test.example.com" and not "_dmarc.example.com")
- o Receivers should quarantine messages from this organizational domain that fail to authenticate ("p=quarantine")
- o Aggregate feedback reports should be sent via email to the addresses "dmarc-feedback@example.com" and "example-tld-test@thirdparty.example.net", with the latter subjected to a maximum size limit ("rua=mailto:dmarc-feedback@example.com,mailto:tld-test@thirdparty.example.net!10m")
- o 25% of messages from this Organizational Domain are subject to action based on this policy ("pct=25")

The DMARC policy record might look like this when retrieved using a common command-line tool (the output shown would appear on a single line, but is wrapped here for publication):

```
% dig +short TXT _dmarc.test.example.com
"v=DMARC1; p=quarantine; rua=mailto:dmarc-feedback@example.com,
mailto:tld-test@thirdparty.example.net!10m; pct=25"
```

To publish such a record, the DNS administrator for the Domain Owner might create an entry like the following in the appropriate zone file:

```
; DMARC record for the domain example.com

_dmarc IN  TXT   ( "v=DMARC1; p=quarantine; "
                  "rua=mailto:dmarc-feedback@example.com, "
                  "mailto:tld-test@thirdparty.example.net!10m; "
                  "pct=25" )
```

B.3. Mail Receiver Example

A Mail Receiver that wants to use DMARC should already be checking SPF and DKIM, and possess the ability to collect relevant information from various email processing stages to provide feedback to Domain Owners.

B.3.1. SMTP-time Processing

An optimal DMARC-enabled Mail Receiver performs authentication and identifier alignment checking during the [[SMTP](#)] conversation.

Prior to returning a final reply to the DATA command, the Mail Receiver's MTA has performed:

1. An SPF check to determine an SPF-authenticated Identifier.
2. DKIM checks that yield one or more DKIM-authenticated Identifiers.
3. A DMARC policy lookup.

The presence of an Author Domain DMARC record indicates that the Mail Receiver should continue with DMARC-specific processing before returning a reply to the DATA command.

Given a DMARC record and the set of Authenticated Identifiers, the Mail Receiver checks to see if the Authenticated Identifiers align with the Author Domain (taking into consideration any "strict" vs "relaxed" options found in the DMARC record).

For example, the following sample data is considered to be from a piece of email originating from the Domain Owner of "example.com":

```
Author Domain: example.com
SPF-authenticated Identifier: mail.example.com
DKIM-authenticated Identifier: example.com
DMARC record:
  "v=DMARC1; p=reject; aspf=r;
  rua=mailto:dmarc-feedback@example.com"
```

In the above sample, both the SPF and the DKIM-authenticated Identifiers align with the Author Domain. The Mail Receiver considers the above email to pass the DMARC check, avoiding the "reject" policy that is to be applied to email that fails to pass the DMARC check.

If no Authenticated Identifiers align with the Author Domain, then the Mail Receiver applies the DMARC-record-specified policy. However, before this action is taken, the Mail Receiver can consult external information to override the Domain Owner's policy. For example, if the Mail Receiver knows that this particular email came from a known and trusted forwarder (that happens to break both SPF and DKIM), then the Mail Receiver may choose to ignore the Domain Owner's policy.

The Mail Receiver is now ready to reply to the DATA command. If the DMARC check yields that the message is to be rejected, then the Mail Receiver replies with a 5xy code to inform the sender of failure. If the DMARC check cannot be resolved due to transient network errors, then the Mail Receiver replies with a 4xy code to inform the sender as to the need to reattempt delivery later. If the DMARC check yields a passing message, then the Mail Receiver continues on with

email processing, perhaps using the result of the DMARC check as an input to additional processing modules such as a domain reputation query.

Before exiting DMARC-specific processing, the Mail Receiver checks to see if the Author Domain DMARC record requests AFRF-based reporting. If so, then the Mail Receiver can emit an AFRF to the reporting address supplied in the DMARC record.

At the exit of DMARC-specific processing, the Mail Receiver captures (through logging or direct insertion into a data store) the result of DMARC processing. Captured information is used to build feedback for Domain Owner consumption. This is not necessary if the Domain Owner has not requested aggregate reports, i.e., no "rua" tag was found in the policy record.

B.4. Utilization of Aggregate Feedback example

Aggregate feedback is consumed by Domain Owners to verify the Domain Owners understanding of how the Domain Owner's Domain is being processed by the Mail Receiver. Aggregate reporting data on emails that pass all DMARC-supporting authentication checks is used by Domain Owners to verify that authentication practices remain accurate. For example, if a third party is sending on behalf of a Domain Owner, the Domain Owner can use aggregate report data to verify ongoing authentication practices of the third party.

Data on email that only partially passes underlying authentication checks provides visibility into problems that need to be addressed by the Domain Owner. For example, if either SPF or DKIM fail to pass, the Domain Owner is provided with enough information to either directly correct the problem or to understand where authentication-breaking changes are being introduced in the email transmission path. If authentication-breaking changes due to email transmission path cannot be directly corrected, then the Domain Owner at least maintains an understanding of the effect of DMARC-based policies upon the Domain Owner's email.

Data on email that fails all underlying authentication checks provides baseline visibility on how the Domain Owner's Domain is being received at the Mail Receiver. Based on this visibility, the Domain Owner can begin deployment of authentication technologies across uncovered email sources. Additionally, the Domain Owner may come to an understanding of how its Domain is being misused.

B.5. mailto Transport example

A DMARC record can contain a "mailto" reporting address, such as:

```
mailto:dmarc-feedback@example.com
```

A sample aggregate report from the Mail Receiver at mail.receiver.example follows:

```
DKIM-Signature: v=1; ...; d=mail.receiver.example; ...
From: dmarc-reporting@mail.receiver.example
Date: Fri, Feb 15 2002 16:54:30 -0800
To: dmarc-feedback@example.com
Subject: Report Domain: example.com
        Submitter: mail.receiver.example
        Report-ID: <2002.02.15.1>
MIME-Version: 1.0
Content-Type: multipart/alternative;
        boundary="-----=_NextPart_000_024E_01CC9B0A.AFE54C00"
Content-Language: en-us
```

This is a multipart message in MIME format.

```
-----=_NextPart_000_024E_01CC9B0A.AFE54C00
Content-Type: text/plain; charset="us-ascii"
Content-Transfer-Encoding: 7bit
```

This is an aggregate report from mail.receiver.example.

```
-----=_NextPart_000_024E_01CC9B0A.AFE54C00
Content-Type: application/gzip
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
        filename="mail.receiver.example!example.com!
                1013662812!1013749130.gz"
```

<gzipped content of report>

```
-----=_NextPart_000_024E_01CC9B0A.AFE54C00--
```

Not shown in the above example is that the Mail Receiver's feedback should be authenticated using SPF. Also, the value of the "filename" MIME parameter is wrapped for printing in this specification but would normally appear as one continuous string.

[Appendix C](#). DMARC XML Schema

The following is the proposed initial schema for producing XML formatted aggregate reports as described in this document.

NOTE: Per the definition of XML, unless otherwise specified in the schema below, the minOccurs and maxOccurs values for each element is set to 1.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://dmarc.org/dmarc-xml/0.1">

  <!-- The time range in UTC covered by messages in this report,
    specified in seconds since epoch. -->
  <xs:complexType name="DateRangeType">
    <xs:all>
      <xs:element name="begin" type="xs:integer"/>
      <xs:element name="end" type="xs:integer"/>
    </xs:all>
  </xs:complexType>

  <!-- Report generator metadata -->
  <xs:complexType name="ReportMetadataType">
    <xs:sequence>
      <xs:element name="org_name" type="xs:string"/>
      <xs:element name="email" type="xs:string"/>
      <xs:element name="extra_contact_info" type="xs:string"
        minOccurs="0"/>
      <xs:element name="report_id" type="xs:string"/>
      <xs:element name="date_range" type="DateRangeType"/>
      <xs:element name="error" type="xs:string" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Alignment mode (relaxed or strict) for DKIM and
    SPF. -->
  <xs:simpleType name="AlignmentType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="r"/>
      <xs:enumeration value="s"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- The policy actions specified by p and sp in the
    DMARC record. -->
  <xs:simpleType name="DispositionType">
```



```
<xs:restriction base="xs:string">
  <xs:enumeration value="none"/>
  <xs:enumeration value="quarantine"/>
  <xs:enumeration value="reject"/>
</xs:restriction>
</xs:simpleType>

<!-- The DMARC policy that applied to the messages in
this report. -->
<xs:complexType name="PolicyPublishedType">
  <xs:all>
    <!-- The domain at which the DMARC record was found. -->
    <xs:element name="domain" type="xs:string"/>
    <!-- The DKIM alignment mode. -->
    <xs:element name="adkim" type="AlignmentType"
      minOccurs="0"/>
    <!-- The SPF alignment mode. -->
    <xs:element name="aspf" type="AlignmentType"
      minOccurs="0"/>
    <!-- The policy to apply to messages from the domain. -->
    <xs:element name="p" type="DispositionType"/>
    <!-- The policy to apply to messages from subdomains. -->
    <xs:element name="sp" type="DispositionType"/>
    <!-- The percent of messages to which policy applies. -->
    <xs:element name="pct" type="xs:integer"/>
    <!-- Failure reporting options in effect. -->
    <xs:element name="fo" type="xs:string"/>
  </xs:all>
</xs:complexType>

<!-- The DMARC-aligned authentication result. -->
<xs:simpleType name="DMARCResultType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="pass"/>
    <xs:enumeration value="fail"/>
  </xs:restriction>
</xs:simpleType>

<!-- Reasons that may affect DMARC disposition or execution
thereof. -->
<xs:simpleType name="PolicyOverrideType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="forwarded"/>
    <xs:enumeration value="sampled_out"/>
    <xs:enumeration value="trusted_forwarder"/>
    <xs:enumeration value="mailing_list"/>
    <xs:enumeration value="local_policy"/>
    <xs:enumeration value="other"/>
  </xs:restriction>
</xs:simpleType>
```



```
</xs:restriction>
</xs:simpleType>

<!-- How do we allow report generators to include new
      classes of override reasons if they want to be more
      specific than "other"? -->
<xs:complexType name="PolicyOverrideReason">
  <xs:all>
    <xs:element name="type" type="PolicyOverrideType"/>
    <xs:element name="comment" type="xs:string"
      minOccurs="0"/>
  </xs:all>
</xs:complexType>

<!-- Taking into account everything else in the record,
      the results of applying DMARC. -->
<xs:complexType name="PolicyEvaluatedType">
  <xs:sequence>
    <xs:element name="disposition" type="DispositionType"/>
    <xs:element name="dkim" type="DMARCResultType"/>
    <xs:element name="spf" type="DMARCResultType"/>
    <xs:element name="reason" type="PolicyOverrideReason"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- Credit to Roger L. Costello for IPv4 regex
      http://mailman.ic.ac.uk/pipermail/xml-dev/1999-December/018018.html -->
<!-- Credit to java2s.com for IPv6 regex
      http://www.java2s.com/Code/XML/XML-Schema/IPv6addressesareeasytodescribeusingasimpleregex.htm -->
<xs:simpleType name="IPAddress">
  <xs:restriction base="xs:string">
    <xs:pattern value="((1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5]).){3}
      (1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])|
      ([A-Fa-f0-9]{1,4}:{1,4}){7}[A-Fa-f0-9]{1,4}"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="RowType">
  <xs:all>
    <!-- The connecting IP. -->
    <xs:element name="source_ip" type="IPAddress"/>
    <!-- The number of matching messages -->
    <xs:element name="count" type="xs:integer"/>
    <!-- The DMARC disposition applying to matching
          messages. -->
```



```
<xs:element name="policy_evaluated"
            type="PolicyEvaluatedType"
            minOccurs="1"/>
</xs:all>
</xs:complexType>

<xs:complexType name="IdentifierType">
  <xs:all>
    <!-- The envelope recipient domain. -->
    <xs:element name="envelope_to" type="xs:string"
                minOccurs="0"/>
    <!-- The envelope from domain. -->
    <xs:element name="envelope_from" type="xs:string"
                minOccurs="1"/>
    <!-- The payload From domain. -->
    <xs:element name="header_from" type="xs:string"
                minOccurs="1"/>
  </xs:all>
</xs:complexType>

<!-- DKIM verification result, according to RFC 5451
Section 2.4.1. -->
<xs:simpleType name="DKIMResultType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="pass"/>
    <xs:enumeration value="fail"/>
    <xs:enumeration value="policy"/>
    <xs:enumeration value="neutral"/>
    <xs:enumeration value="temperror"/>
    <xs:enumeration value="permerror"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="DKIMAuthResultType">
  <xs:all>
    <!-- The d= parameter in the signature -->
    <xs:element name="domain" type="xs:string"
                minOccurs="1"/>
    <!-- The s= parameter in the signature -->
    <xs:element name="selector" type="xs:string"
                minOccurs="0"/>
    <!-- The DKIM verification result -->
    <xs:element name="result" type="DKIMResultType"
                minOccurs="1"/>
    <!-- Any extra information (e.g., from
         Authentication-Results -->
    <xs:element name="human_result" type="xs:string"
```



```
        minOccurs="0"/>
    </xs:all>
</xs:complexType>

<!-- SPF domain scope -->
<xs:simpleType name="SPFDomainScope">
    <xs:restriction base="xs:string">
        <xs:enumeration value="helo"/>
        <xs:enumeration value="mfrom"/>
    </xs:restriction>
</xs:simpleType>

<!-- SPF result -->
<xs:simpleType name="SPFResultType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="none"/>
        <xs:enumeration value="neutral"/>
        <xs:enumeration value="pass"/>
        <xs:enumeration value="fail"/>
        <xs:enumeration value="softfail"/>
        <!-- "TempError" commonly implemented as "unknown" -->
        <xs:enumeration value="temperror"/>
        <!-- "PermError" commonly implemented as "error" -->
        <xs:enumeration value="permerror"/>
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="SPFAuthResultType">
    <xs:all>
        <!-- The checked domain. -->
        <xs:element name="domain" type="xs:string" minOccurs="1"/>
        <!-- The scope of the checked domain. -->
        <xs:element name="scope" type="SPFDomainScope" minOccurs="1"/>
        <!-- The SPF verification result -->
        <xs:element name="result" type="SPFResultType"
            minOccurs="1"/>
    </xs:all>
</xs:complexType>

<!-- This element contains DKIM and SPF results, uninterpreted
    with respect to DMARC. -->
<xs:complexType name="AuthResultType">
    <xs:sequence>
        <!-- There may be no DKIM signatures, or multiple DKIM
            signatures. -->
        <xs:element name="dkim" type="DKIMAuthResultType"
            minOccurs="0" maxOccurs="unbounded"/>
        <!-- There will always be at least one SPF result. -->
```



```
<xs:element name="spf" type="SPFAuthResultType" minOccurs="1"
  maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<!-- This element contains all the authentication results that
  were evaluated by the receiving system for the given set of
  messages. -->
<xs:complexType name="RecordType">
  <xs:sequence>
    <xs:element name="row" type="RowType"/>
    <xs:element name="identifiers" type="IdentifierType"/>
    <xs:element name="auth_results" type="AuthResultType"/>
  </xs:sequence>
</xs:complexType>

<!-- Parent -->
<xs:element name="feedback">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="version"
        type="xs:decimal"/>
      <xs:element name="report_metadata"
        type="ReportMetadataType"/>
      <xs:element name="policy_published"
        type="PolicyPublishedType"/>
      <xs:element name="record" type="RecordType"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Descriptions of the PolicyOverrideTypes:

forwarded: Message was relayed via a known forwarder, or local heuristics identified the message as likely having been forwarded. There is no expectation that authentication would pass.

local_policy: The Mail Receiver's local policy exempted the message from being subjected to the Domain Owner's requested policy action.

mailing_list: Local heuristics determined that the message arrived via a mailing list, and thus authentication of the original message was not expected to succeed.

other: Some policy exception not covered by the other entries in this list occurred. Additional detail can be found in the PolicyOverrideReason's "comment" field.

sampled_out: Message was exempted from application of policy by the "pct" setting in the DMARC policy record.

trusted_forwarder: Message authentication failure was anticipated by other evidence linking the message to a locally-maintained list of known and trusted forwarders.

The "version" for reports generated per this specification MUST be the value 1.0.

Appendix D. Public Discussion

Public discussion of the DMARC proposal documents is taking place on the dmarc-discuss@dmarc.org mailing list. Subscription is available at <http://www.dmarc.org/mailman/listinfo/dmarc-discuss>.

[RFC Editor: Please remove this section prior to publication.]

Appendix E. Acknowledgements

DMARC and the version of this document submitted to the IETF were the result of lengthy efforts by an informal industry consortium: DMARC.org [1]. Participating companies included: Agari, American Greetings, AOL, Bank of America, Cloudmark, Comcast, Facebook, Fidelity Investments, Google, JPMorgan Chase & Company, LinkedIn, Microsoft, Netease, Paypal, ReturnPath, Trusted Domain Project, and Yahoo!. Although the number of contributors and supporters are too numerous to mention, notable individual contributions were made by J. Trent Adams, Michael Adkins, Monica Chew, Dave Crocker, Tim Draegen, Steve Jones, Franck Martin, Brett McDowell, and Paul Midgen. The contributors would also like to recognize the invaluable input and guidance that was provided early on by J.D. Falk.

Additional contributions within the IETF context were made by Les Barstow, Jim Fenton, J. Gomez, Mike Jones, Scott Kitterman, Eliot Lear, John Levine, S. Moonesamy, Henry Timmes, and Stephen J. Turnbull.

Authors' Addresses

Murray S. Kucherawy (editor)

Email: superuser@gmail.com

Elizabeth Zwicky (editor)

Yahoo!

Email: zwicky@yahoo-inc.com