Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 25, 2019

T. Sirainen
Open-Xchange Oy
M. Slusarz
Open-Xchange Inc.

March 24, 2019

# SMTP: Submission Token Extension
# draft-slusarz-extra-smtp-submission-token-00

Abstract

This document specifies an extension to a SMTP submission server that allows synchronous message delivery via use of a limited-privilege authentication token.

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 https://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 September 25, 2019.

Table of Contents

## 1. Introduction

Electronic mail is a widely-used messaging system that allows any user to join the network if they conform to the open standards defining the federation.

This open federation is one of the strengths of the design, and is responsible for much of the medium's popularity. This open federation ensures that no one organization controls the platform. However, this lack of centralization has required substantial network engineering improvements over time to ensure that mails can be routed and delivered, as there is no guarantee that delivery paths will remain stable and reliable over time.

This patchwork of systems, layered on top of legacy SMTP [RFC5321] has done a remarkable job in practice of ensuring reliable mail delivery. However, an artifact of these systems is that latency of mail message delivery is highly variable, as routings designed to handle all potential use cases may result in very convoluted delivery paths for a subset of messages.

Since email was first created, new messaging paradigms have arisen that allow for an improved real-time message delivery experience. In order to keep email relevant with current user expectations, it is desirable to improve delivery speeds to match these new technologies.

This extension provides a solution to allow more real-time message delivery capabilities to email through minimal backward-compatible changes to the legacy SMTP paradigm. This allows for a messaging experience that can be incrementally rolled out to the system - falling back to the existing, reliable email delivery network if not available - and has no visible effect on an end user other than decreased delivery times for the portion of their email traffic using the new standard.

Secondarily, this extension also improves security of mail transfer by requiring encrypted network connections, leveraging existing standards, to transfer mail messages. This aligns with the general trend in messaging systems of encrypting in-transit data as much as possible.

## 2. Conventions Used In This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

"User" is used to refer to a human user, whereas "client" refers to the software being run by the user.

"Token Server" refers to the submission server where authentication occurs and messages are delivered. "Remote Client" is the software that possesses a token, generated by the Token Server, and authenticates to the Token Server with that token to deliver a message.

In examples, "C:" and "S:" indicate lines sent by the client and server respectively. If a single "C:" or "S:" label applies to multiple lines, then the line breaks between those lines are for editorial clarity only and are not part of the actual protocol exchange.

TODO: note on case insensitivity of protocol examples

## 3. The Submission Token Service Extension

1. The name of this SMTP [RFC5321] service extension is "Submission Token".

2. The EHLO keyword value associated with this extension is "STOKEN".

3. Two new SMTP [RFC5321] verbs are defined: "GENSTOKEN" and "REVSTOKEN"

4. Two optional parameters, using the keywords "STOKEN" and "MYSTOKEN", are added to the RCPT TO command.

5. This extension is appropriate, and exclusive to, the submission protocol [RFC6409].

6. This extension is appropriate, and exclusive to, Local Mail Transfer Protocol (LMTP) [RFC2033].

7. When this extension is used, the trace field MUST use the WITH keyword LMTPSA [RFC3848].

8. When this extension is used, the DNS Submission SRV [RFC6186] record MUST be specified.

9. When this extension is used, the ENHANCEDSTATUSCODES [RFC2034] service extension MUST be available.

## 4. Overview

A brief overview of how submission tokens work is as follows:

1. A temporary token is generated on the Token Server (See Section 5)

2. The token is distributed to the remote recipient(s) (See Section 6)

3. The remote recipient, via a Remote Client, connects to the Token Server, via the submission protocol, and authenticates using the token. Once authenticated, the Remote Client uses LMTP to deliver the message.

   * During delivery, a temporary token is replaced with a new permanent token issued by the Token Server.

   * (Optional) The Remote Client provides its permanent token to the Token Server (if it can handle incoming submission connections) to allow the reverse delivery flow to use token authentication.

The remaining sections describe the details of this process.

## 5. Token

### 5.1. Description

Tokens are a variable-length string, dynamically generated by a Token Server, that are used to allow a specific remote recipient to authenticate to the Token Server for the limited purpose of delivering a message to the owner of the token.

All tokens MUST be tied to a specific pair of recipients (remote and local). All tokens SHOULD be time limited; the duration is dictated by the token type (see below).

### 5.2. Format

The token format is not defined by this document and is server-dependent.

TODO: Should token format be standardized

### 5.3. Types

Two categories of tokens are defined: temporary and permanent.

#### 5.3.1. Temporary

A temporary token is designed to be created by the Token Server for an initial engagement with a remote recipient.

Temporary tokens are expected to be distributed through possible non-secure channels. Therefore, sessions authenticated via a temporary token MAY be treated as less trusted.

Temporary tokens SHOULD be time-limited to a valid duration of one week.

The token MAY be used multiple times, as a remote recipient may use multiple Token Clients. Thus, temporary token validity SHOULD be tied to time limitations, not use limitations.

Temporary tokens can be generated in a manner whereby the token isn't stored by the Token Server. Instead, these tokens can be created such that a Token Server can verify whether the token is valid based solely on the contents of the token string. This allows generation of large numbers of temporary tokens, some that may never be used, without imposing a storage burden on the Token Server.

TODO: Provide example of self-verifying token

#### 5.3.2. Permanent

A permanent token is generated when a Remote Client successfully delivers a message using a temporary token. This delivery (along with TLS verification required as part of the delivery process) provides a measure of trust verification regarding the recipient to recipient delivery chain, such that future use of a permanent token to deliver messages SHOULD be treated as a more trusted delivery than non-permanent token deliveries.

The permanent token is intended to be only transferred over a secure channel, unlike a temporary token.

Since trust in a permanent token is higher than a temporary token, the expiration date of the token can be significantly longer. A permanent token SHOULD be time-limited to a valid duration of one year.

In order to maintain the trusted relationship, once established, a server MAY refresh or extend a permanent token during any successful delivery. It is server-dependent when this occurs. A Remote Client MUST expect and handle a token regeneration if a server provides a new permanent token.

## 5.4. Management

Token generation MAY be transparently handled by a Token Server. However, token management must also be accessible to a connecting client. This subsection defines two SMTP commands that allow this management.

### 5.4.1. GENSTOKEN (Generate Submission Token) Command

GENSTOKEN ( "PERM" / "TEMP" ) remote-address [local-address]

Generates a token for given local/remote address pair.

If permanent token already exists, return that token.

On success, return 250 reply code, with a 2.1.11 extended status code; the response contains the generated token.

On error: If remote address is incorrect, return 501 reply code with a 5.1.3 extended status code. If local address is provided and is incorrect, return 501 reply code with a 5.1.7 extended status code. If token could not be generated for the remote/local address pair, return 451 reply code with a 4.5.0 extended status code.

TODO: local-address optional

TODO: If temporary token already exists, generate a new one?

TODO: Does generation of already existing permanent token count as a refresh/renew, and a new one is generated?

#### 5.4.1.1. Enhanced Status Code on Successful Token generation

| | |
|---|---|
| Code: | X.1.11 |
| Sample Text: | Submission Token Successfully Created |
| Associated basic status code: | 250 |
| Description: | This status code is returned when a submission token is successfully generated with the GENSTOKEN command. The humantext associated with this code has a defined format of: token humantext |

#### 5.4.1.2. Examples

Example: Creating a temporary token using GENSTOKEN

```
C: GENSTOKEN TEMP user@example.com localuser@foo.com
S: 250 2.1.11 17yUEeUu8M Temporary token generated.
```

Example: GENSTOKEN Error (incorrect remote address)

```
C: GENSTOKEN TEMP remoteuser..@example.com
S: 501 5.1.3 Incorrect remote address.
```

### 5.4.2. REVSTOKEN (Revoke Submission Token) Command

REVSTOKEN remote-address [local-address]

Revokes all tokens for a given local/remote address pair.

On success, return 250 reply code, with a 2.1.0 extended status code. If no tokens exist for a local/remote address pair, this should be treated as a successful command completion.

On error: If remote address is incorrect, return 501 reply code with a 5.1.3 extended status code. If local address is provided and is incorrect, return 501 reply code with a 5.1.7 extended status code. If token could not be revoked for the remote/local address pair, return 451 reply code with a 4.5.0 extended status code.

TODO: local-address optional

TODO: Is success returned even if token does not exist?

#### 5.4.2.1. Examples

Example: Revoking tokens using REVSTOKEN

```
C: REVSTOKEN user@example.com localuser@foo.com
S: 250 2.1.0 All tokens successfully revoked.
```

Example: REVSTOKEN Error (incorrect remote address)

```
C: REVSTOKEN remoteuser..@example.com
S: 501 5.1.3 Incorrect remote address.
```

# 6. Distribution

## 6.1. Description

Temporary token distribution to an external recipient can be accomplished in any manner deemed appropriate by an implementer.

This document defines a method that temporary tokens can be distributed to a recipient via email headers.

## 6.2. Email Header Distribution

Temporary tokens can be distributed by use of a newly-defined Submission-Token email header.

Submission server can automatically generate tokens for outgoing messages

Submission-Token: <token>

There can be multiple Submission-Token headers, as each contact can have multiple aliases.

TODO: Add submission host hint?

TODO: RFC mail header definition requirements?

TODO: Define token format?

# 7. Message Delivery Using Submission Token

## 7.1. Summary

This section summarizes the process for a Remote Client to deliver a message by connecting to the Token Server with a submission token valid for the sending user and the message recipient.

If the Remove Client possesses a valid token for the sender/recipient pair, it can attempt to use submission token routing to deliver the message. Otherwise, the Remote Client can fallback to sending the message through basic submission deliver behavior.

## 7.2. Connection

The Remote Client connects to the Token Server.

The Token Server for a recipient is discovered via the DNS SRV [RFC6186] submission record.

TODO: Use full domain, then fallback to base domain?

If the DNS SRV submission record doesn't exist, or the host cannot be contacted, a Remote Client MAY fallback to using the server address provided by the DNS MX [RFC5321] record.

If Remote Client cannot determine the address for the Token Server, the message should be sent via normal mail submission [RFC6409] channels.

Once the Token Server address is determined, and a connection can be established to that address, this connection MUST be protected by TLS. The mechanism and port used to enable this secure connection is described in RFC 8314 [RFC8314]. Continuing discussion here assumes a TLS layer has been established based on the requirements contained in RFC 8314.

STOKEN MUST NOT be listed as a service extension on the Token Server until TLS is active.

TODO: TLS domain/certificate check a MUST?

### 7.3. Authentication

Once connected, and TLS is active, the client MUST send a LHLO command. This command indicates that the protocol used is LMTP [RFC2033]. LMTP is used because submission token-based delivery requires guaranteed synchronous message delivery where the message MUST NOT be queued.

The service extension identifier STOKEN must be output as part of the return from the LHLO command. STOKEN must not be announced as a return from the HELO or EHLO commands. If STOKEN is not listed, the Remote Client should immediately terminate the submission sessions and proceed to delivering the message through basic submission delivery.

Authentication to access the Token Server with a submission token is accomplished via the AUTH [RFC4954] command.

#### 7.3.1. SASL Extension (TODO)

TODO: Need to define SASL [RFC4422] method to do authentication. Idea: "AUTH STOKEN input", where input is a base64 [RFC4648] encoded string of "recipient\0stoken". Recipient is used as a simple DoS prevention, requiring that a submission token alone is not enough to access the server.

If multiple recipients exist on the Token Server, and Remote Client possesses multiple submission tokens, any single token valid for a recipient/token pair that the Token Server can service will suffice to pass the authentication check.

See https://tools.ietf.org/html/rfc4422#section-5 for requirements in defining new SASL mechanism.

See https://tools.ietf.org/html/rfc4505#section-2 for example SASL mechanism definition

"normalization"? is foo@server.example.com the same as foo@example.com?

#### 7.3.1.1. Examples

Example: SASL STOKEN Auth

```
C: AUTH STOKEN dXNlckBmb28uY29tXDBzNnhXMTVoOURo
S: 235 2.7.0 OK
```

Example: SASL STOKEN Auth Error

```
C: AUTH STOKEN abcdefg
S: 535 5.7.8 Could not authenticate.
```

### 7.4. Delivery

Delivery is restricted to users for which the Remote Client has a valid submission token and the message can be synchronously delivered. The token for a given recipient is specified via the STOKEN parameter to the RCPT TO command.

If the Remote Client supports submission token delivery, the Remote Client can provide a permanent token to the Token Server via the MYSTOKEN parameter to the RCPT TO command. This allows the Token Server recipient to be able to connect back to the Remote Client submission server to deliver messages in the future.

#### 7.4.1. Enhanced Status Codes on Successful Delivery

#### 7.4.1.1. Successful Delivery with Delivery ID

It can be useful for a client to record a delivery tracking ID provided by the submission server after a successful delivery. Many submission servers add this information to the humantext that follows the status code and optional enhanced status code. However, there is

no current standard that exists regarding the formatting of this info across disparate submission implementations.

This extension adds an enhanced status code that defines the format of the humantext after a successful delivery to allow for reliable machine-automated extraction of this delivery ID.

| | |
|---|---|
| Code: | X.1.12 |
| Sample Text: | Message successfully delivered |
| Associated basic status code: | 250 |
| Description: | This status code is returned when a message is successfully delivered, and a unique delivery ID has been generated and is being provided to the sender. The humantext associated with this code has a defined format of:<br><delivery-address> delivery-id humantext |

#### 7.4.1.2. Successful Delivery with Delivery ID and Permanent Token Exchange

In addition to a delivery ID, upon a successfully delivery a Token Server may either need to 1) replace a temporary token with a permanent token, or 2) replace an existing permanent token with a new permanent token. This information is broadcast to the Remote Client by means of structured data presented in the 250 success response to the DATA command.

This extension adds an enhanced status code that defines the format of the humantext after a successful delivery to allow for reliable machine-automated extraction of both the delivery ID code and the permanent token.

| | |
|---|---|
| Code: | X.1.13 |
| Sample Text: | Message successfully delivered and permanent submission token generated |
| Associated basic status code: | 250 |
| Description: | This status code is returned when a message is successfully delivered and a unique delivery ID has been generated and is being provided to the sender. The humantext associated with this code has a defined format of:<br><delivery-address> token delivery-id humantext |

## 8. Examples

Example 1: Remote Client using a temporary token to deliver a message via a Token Server

```
[...TLS layer negotiated...]
C: LHLO sender.example.com
S: 250-foo.com
S: 250-ENHANCEDSTATUSCODES
S: 250 STOKEN
C: AUTH STOKEN dXNlckBmb28uY29tXDBzNnhXMTVoOURo
S: 235 2.7.0 OK
C: MAIL FROM:<user@example.com>
S: 250 2.1.0 OK
C: RCPT TO:<user@foo.com> STOKEN=s6xW15h9Dh MYSTOKEN=Enm3HX76Mb
S: 250 2.1.5 OK
C: RCPT TO:<user2@foo.com> STOKEN=mfpWamXp5L MYSTOKEN=H4tGNfh6Us
S: 250 2.1.5 OK
C: DATA
S: 354 OK
C: [...message data...]
C: .
S: 250 2.1.13 <user@foo.com> etLBdTj1iG NRaaVe83QN Saved
S: 250 2.1.13 <user2@foo.com> bS6zMW8Hrk VddCEQDVyp Saved
```

Example 2: Remote Client using permanent token to deliver a message via a Token Server

```
C: LHLO sender.example.com
S: 250-foo.com
S: 250-ENHANCEDSTATUSCODES
S: 250 STOKEN
C: AUTH STOKEN dXNlckBmb28uY29tXDB6NEpIbVVOSks0
S: 235 2.7.0 OK
C: MAIL FROM:<user@example.com>
S: 250 2.1.0 OK
C: RCPT TO:<user@foo.com> STOKEN=z4JHmUNJK4 MYSTOKEN=YlcDrrLZEC
S: 250 2.1.5 OK
C: RCPT TO:<user2@foo.com> STOKEN=m6Cn9lIKjU MYSTOKEN=kAT0G96njA
S: 250 2.1.5 OK
C: DATA
S: 354 OK
C: [...message data...]
C: .
S: 250 2.1.12 <user@foo.com> imXtPChKDO Saved
S: 250 2.1.13 <user2@foo.com> CXJSmLdbeh 2WXuXmKG15 Saved
   and new token returned
```

Example 3: TODO - Error examples

## 9. Formal Syntax

The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in ABNF [RFC5234]. It includes definitions from SMTP [RFC5321].

Except as noted otherwise, all alphabetic characters are case-insensitive. The use of upper or lower case characters to define token strings is for editorial clarity only. Implementations MUST accept these strings in a case-insensitive fashion.

```
esmtp-keyword /=  "STOKEN" / "MYSTOKEN"

token          =  string  ; TODO

delivery-id    =  string  ; TODO
```

## 10. IANA Considerations

### 10.1. SMTP Extension Registration

Section 2.2.2 of SMTP [RFC5321] defines the the procedure for registering a new SMTP extension. It is requested that IANA registers the new SMTP extension STOKEN using the details provided in Section 3 of this document.

### 10.2. Enhanced Status Code Registration

Section 2.3 of RFC 5248 [RFC5248] defines the procedure for registering a new enumerated status code in the "Simple Mail Transfer Protocol (SMTP) Enhanced Status Codes Registry". It is requested that IANA registers the new enhanced status codes using the details provided in Section 5.4.1.1 and Section 7.4.1 of this document.

## 11. Security Considerations

TODO

SPF/DKIM

Remote authentication

Storage of tokens

Interception of temporary token.

DoS.

SPAM - still need to scan!

Token use increases trust, but does not guarantee it.

## 12. References

### 12.1. Normative References

[RFC2033] Myers, J., "Local Mail Transfer Protocol", RFC 2033, DOI 10.17487/RFC2033, October 1996, <https://www.rfc-editor.org/info/rfc2033>.

[RFC2034] Freed, N., "SMTP Service Extension for Returning Enhanced Error Codes", RFC 2034, DOI 10.17487/RFC2034, October 1996, <https://www.rfc-editor.org/info/rfc2034>.

[RFC3848] Newman, C., "ESMTP and LMTP Transmission Types Registration", RFC 3848, DOI 10.17487/RFC3848, July 2004, <https://www.rfc-editor.org/info/rfc3848>.

[RFC4422] Melnikov, A., Ed. and K. Zeilenga, Ed., "Simple Authentication and Security Layer (SASL)", RFC 4422, DOI 10.17487/RFC4422, June 2006, <https://www.rfc-editor.org/info/rfc4422>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <https://www.rfc-editor.org/info/rfc5321>.

[RFC6186] Daboo, C., "Use of SRV Records for Locating Email Submission/Access Services", RFC 6186, DOI 10.17487/RFC6186, March 2011, <https://www.rfc-editor.org/info/rfc6186>.

[RFC6409] Gellens, R. and J. Klensin, "Message Submission for Mail", STD 72, RFC 6409, DOI 10.17487/RFC6409, November 2011, <https://www.rfc-editor.org/info/rfc6409>.

[RFC8314] Moore, K. and C. Newman, "Cleartext Considered Obsolete: Use of Transport Layer Security (TLS) for Email Submission and Access", RFC 8314, DOI 10.17487/RFC8314, January 2018, <https://www.rfc-editor.org/info/rfc8314>.

## 12.2. Informative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC4954] Siemborski, R., Ed. and A. Melnikov, Ed., "SMTP Service Extension for Authentication", RFC 4954, DOI 10.17487/RFC4954, July 2007, <https://www.rfc-editor.org/info/rfc4954>.

[RFC5248] Hansen, T. and J. Klensin, "A Registry for SMTP Enhanced Mail System Status Codes", BCP 138, RFC 5248, DOI 10.17487/RFC5248, June 2008, <https://www.rfc-editor.org/info/rfc5248>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

## Appendix A. Change History (To be removed by RFC Editor before publication)

## Acknowledgments

The authors would like to thank the following people for their comments and contributions to this document: TODO.

## Authors' Addresses

Timo Sirainen
Open-Xchange Oy
Lars Sonckin kaari 12
Espoo 02600
FI

Email: timo.sirainen@open-xchange.com

Michael M. Slusarz
Open-Xchange Inc.
530 Lytton Avenue
Palo Alto, California 94301
US

Email: michael.slusarz@open-xchange.com