Sieve Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 5, 2009

A. Melnikov, Ed.
Isode Limited
T. Martin
BeThereBeSquare Inc.

January 1, 2009

A Protocol for Remotely Managing Sieve Scripts
draft-ietf-sieve-managesieve-06

Status of this Memo

Copyright Notice

Abstract

Sieve scripts allow users to filter incoming email. Message stores

are commonly sealed servers so users cannot log into them, yet users must be able to update their scripts on them. This document describes a protocol "ManageSieve" for securely managing Sieve scripts on a remote server. This protocol allows a user to have multiple scripts, and also alerts a user to syntactically flawed scripts.

Changes since draft-martin-managesieve-09

- TBD.

Table of Contents

1. Introduction

1.1. Conventions used in this 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].

In examples, "C:" and "S:" indicate lines sent by the client and server respectively. Line breaks that do not start a new "C:" or "S:" exist for editorial reasons.

1.2. Commands and Responses

A ManageSieve connection consists of the establishment of a client/server network connection, an initial greeting from the server, and client/server interactions. These client/server interactions consist of a client command, server data, and a server completion result response.

All interactions transmitted by client and server are in the form of lines, that is, strings that end with a CRLF. The protocol receiver of a ManageSieve client or server is either reading a line, or is reading a sequence of octets with a known count followed by a line.

1.3. Syntax

This a line oriented protocol much like [IMAP] or [ACAP]. There are three data types: atoms, numbers and strings. Strings may be quoted or literal. See [ACAP] for detailed descriptions of these types.

Each command consists of an atom (the command name) followed by zero or more strings and numbers terminated by CRLF.

All client queries are replied to with either an OK, NO, or BYE response. Each response may be followed by a response code (see Section 1.4) and by a string consisting of human readable text in the local language (as returned by the LANGUAGE capability, see Section 1.8), encoded in [UTF-8]. The contents of the string SHOULD be shown to the user and implementations MUST NOT attempt to parse the message for meaning.

The BYE response SHOULD be used if the server wishes to close the connection. A server may wish to do this because the client was idle for too long or there were too many failed authentication attempts. This response can be issued at any time and should be immediately followed by a server hang-up of the connection. If a server has an inactivity timeout resulting in client autologout it MUST be no less

than 30 minutes after successful authentication. The inactivity timeout MAY be less before authentication.

1.4. Response Codes

An OK, NO, or BYE response from the server MAY contain a response code to describe the event in a more detailed machine parsable fashion. A response code consists of data inside parentheses in the form of an atom, possibly followed by a space and arguments. Response codes are defined when there is a specific action that a client can take based upon the additional information. In order to support future extension, the response code is represented as a slash-separated (Solidus, %x2F) hierarchy with each level of hierarchy representing increasing detail about the error. Response codes MUST NOT start with the Solidus character. Clients MUST tolerate additional hierarchical response code detail which they don't understand. For example, if the client supports the "QUOTA" response code, but doesn't understand the "QUOTA/MAXSCRIPTS" response code, it should treat "QUOTA/MAXSCRIPTS" as "QUOTA".

Client implementations MUST tolerate (ignore) response codes that they do not recognize.

The currently defined response codes are:

AUTH-TOO-WEAK

This response code is returned in the NO or BYE response from an AUTHENTICATE command. It indicates that site security policy forbids the use of the requested mechanism for the specified authentication identity.

ENCRYPT-NEEDED

This response code is returned in the NO or BYE response from an AUTHENTICATE command. It indicates that site security policy requires the use of a strong encryption mechanism for the specified authentication identity and mechanism.

QUOTA

If this response code is returned in the NO/BYE response, it means that the command would have placed the user above the site-defined quota constraints. If this response code is returned in the OK response, it can mean that the user's storage is near its quota, or it can mean that the account exceeded its quota but that that condition is being allowed by the server (the server supports so called "soft quotas"). The QUOTA response code has 2 more detailed

variants: "QUOTA/MAXSCRIPTS" (the maximum number of per-user scripts) and "QUOTA/MAXSIZE" (the maximum script size).

REFERRAL

This response code may be returned with a BYE result from any command, and includes a mandatory parameter that indicates what server to access to manage this user's sieve scripts. The server will be specified by a Sieve URL (see Section 3). The scriptname portion of the URL MUST NOT be specified. The client should authenticate to the specified server and use it for all further commands in the current session.

SASL

This response code can occur in the OK response to a successful AUTHENTICATE command and includes the optional final server response data from the server as specified by [SASL].

TRANSITION-NEEDED

This response code occurs in a NO response of an AUTHENTICATE command. It indicates that the user name is valid, but the entry in the authentication database needs to be updated in order to permit authentication with the specified mechanism. This is typically done by establishing a secure channel using TLS, verifying server identity as specified in Section 2.2.1, and finally authenticating once using the [PLAIN] authentication mechanism. The selected mechanism SHOULD then work for authentications in subsequent sessions.

This condition can happen if a user has an entry in a system authentication database such as Unix /etc/passwd, but does not have credentials suitable for use by the specified mechanism.

TRYLATER

A command failed due to a temporary server failure. The client MAY continue using local information and try the command later. This response code only makes sense when returned in a NO/BYE response.

ACTIVE

A command failed because it is not allowed on the active script. For example DELETESCRIPT on the active script. This response code only makes sense when returned in a NO/BYE response.

NONEXISTENT

A command failed because the referenced script name doesn't exist. This response code only makes sense when returned in a NO/BYE response.

ALREADYEXISTS

A command failed because the referenced script name already exists. This response code only makes sense when returned in a NO/BYE response.

TAG

This response code name is followed by a string specified in the command. See Section 2.13 for a possible use case.

WARNINGS

This response code MAY be returned by the server in the OK response (but it might be returned with the NO/BYE response as well) and signals the client that even though the script is syntactically valid, it might contain errors not intended by the script writer. This response code is typically returned in response to PUTSCRIPT and/or CHECKSCRIPT commands. A client seeing such response code SHOULD present the returned warning text to the user.

1.5. Active Script

A user may have multiple Sieve scripts on the server, yet only one script may be used for filtering of incoming messages. This is the active script. Users may have zero or one active scripts and MUST use the SETACTIVE command described below for changing the active script or disabling Sieve processing. For example, a user may have an everyday script they normally use and a special script they use when they go on vacation. Users can change which script is being used without having to download and upload a script stored somewhere else.

1.6. Quotas

Servers SHOULD impose quotas to prevent malicious users from overflowing available storage. If a command would place a user over a quota setting, servers that impose such quotas MUST reply with a NO response containing the QUOTA response code. Client implementations MUST be able to handle commands failing because of quota restrictions.

1.7. Script Names

A Sieve script name is a sequence of Unicode characters encoded in UTF-8 [UTF-8]. A script name MUST comply with Net-Unicode Definition (Section 2 of [NET-UNICODE]), with the additional restriction of prohibiting the following Unicode characters:

- 0000-001F; [CONTROL CHARACTERS]
- 007F; DELETE
- 0080-009F; [CONTROL CHARACTERS]
- 2028; LINE SEPARATOR
- 2029; PARAGRAPH SEPARATOR

Sieve script names MUST be at least one octet (and hense Unicode character) long. Zero octets script name has a special meaning (see Section 2.8). Servers MUST allow names of up to 128 Unicode characters in length (which can take up to 512 bytes when encoded in UTF-8, not counting the terminating NUL), and MAY allow longer names. A server that receives a script name longer than its internal limit MUST reject the corresponding operation, in particular it MUST NOT truncate the script name.

1.8. Capabilities

Server capabilities are sent automatically by the server upon a client connection, or after successful STARTTLS and AUTHENTICATE (which establishes a SASL security layer) commands. Capabilities may change immediately after a successfully completed STARTTLS command, and/or immediately after a successfully completed AUTHENTICATE command, and/or after a successfully completed UNAUTHENTICATE command (see Section 2.14.1). Capabilities MUST remain static at all other times.

Clients MAY request the capabilities at a later time by issuing the CAPABILITY command described later. The capabilities consist of a series of lines each with one or two strings. The first string is the name of the capability, which is case-insensitive. The second optional string is the value associated with that capability. Order of capabilities is arbitrary, but each capability name can appear at most once.

The following capabilities are defined in this document:

IMPLEMENTATION - Name of implementation and version. This capability

MUST always be returned by the server.

SASL - List of SASL mechanisms supported by the server, each separated by a space. This list can be empty if and only if STARTTLS is also advertised. This means that the client must negotiate TLS encryption with STARTTLS first, at which point the SASL capability will list a non empty list of SASL mechanisms.

SIEVE - List of space separated Sieve extensions (as listed in Sieve "require" action [SIEVE]) supported by the Sieve engine. This capability MUST always be returned by the server.

STARTTLS - If TLS [TLS] is supported by this implementation. Before advertising this capability a server MUST verify to the best of its ability that TLS can be successfully negotiated by a client with common cipher suites. Specifically, a server should verify that a server certificate has been installed and that the TLS subsystem has successfully initialized. This capability SHOULD NOT be advertised once STARTTLS or AUTHENTICATE command completes successfully. Client and server implementations MUST implement the STARTTLS extension.

MAXREDIRECTS - Specifies the limit on the number of Sieve "redirect" actions a script can perform during a single evaluation. Note, that this is different from the total number of "redirect" actions a script can contain. The value is a non-negative number represented as a ManageSieve string.

NOTIFY - A space separated list of URI schema parts for supported notification methods. This capability MUST be specified if the Sieve implementation supports the "enotify" extension [NOTIFY].

LANGUAGE - The language (<Language-Tag> from [RFC4646]) currently used for human readable error messages. If this capability is not returned, the "i-default" [RFC2277] language is assumed. Note that the current language MAY be per-user configurable (i.e. it MAY change after authentication).

OWNER - The canonical name of the logged in user (SASL "authorization identity") encoded in UTF-8. This capability MUST NOT be returned in unauthenticated state and SHOULD be returned once the AUTHENTICATE command succeeds.

VERSION - This capability MUST be returned by servers compliant with this document or its successor. For servers compliant with this document the capability value is the string "1.0". Lack of this capability means that the server predates this specification and thus doesn't support the following commands: RENAMESCRIPT, CHECKSCRIPT and NOOP.

Section 2.14 defines some additional ManageSieve extensions and their respective capabilities.

A server implementation MUST return SIEVE, IMPLEMENTATION and VERSION capabilities.

A client implementation MUST ignore any listed capabilities that it does not understand.

Example:

```
S: "IMPlemENTATION" "Example1 ManageSieved v001"
S: "SASl" "DIGEST-MD5 GSSAPI"
S: "SIeVE" "fileinto vacation"
S: "StaRTTLS"
S: "NOTIFY" "xmpp mailto"
S: "MAXREdIRECTS" "5"
S: "VERSION" "1.0"
S: OK
```

After successful authentication this might look like this:

Example:

```
S: "IMPlemENTATION" "Example1 ManageSieved v001"
S: "SASl" "DIGEST-MD5 GSSAPI"
S: "SIeVE" "fileinto vacation"
S: "NOTIFY" "xmpp mailto"
S: "OWNER" "alexey@example.com"
S: "MAXREdIRECTS" "5"
S: "VERSION" "1.0"
S: OK
```

1.9. Link Level

The ManageSieve protocol assumes a reliable data stream such as that provided by TCP. When TCP is used, a ManageSieve server typically listens on port 2000. [[anchor7: IANA registration of port 2000 is pending.]]

Before opening the TCP connection, the ManageSieve client first MUST resolve the Domain Name System (DNS) hostname associated with the receiving entity and determine the appropriate TCP port for communication with the receiving entity. The process is as follows:

1. Attempt to resolve the hostname using a [DNS-SRV] Service of "sieve" and a Proto of "tcp" for the target domain (e.g. "example.net"), resulting in resource records such as

"_sieve._tcp.example.net.". The result of the SRV lookup, if successful, will be one or more combinations of a port and hostname; the ManageSieve client MUST resolve the returned hostnames to IPv4/IPv6 addresses according to returned SRV record weight. IP addresses from the first successfully resolved hostname (with the corresponding port number returned by SRV lookup) are used to connect to the server. If connection using one of the IP addresses fails, the next resolved IP address is used to connect. If connection to all resolved IP addresses fails, then the resolution/connect is repeated for the next hostname returned by SRV lookup.

2. If the SRV lookup fails, the fallback SHOULD be a normal IPv4 or IPv6 address record resolution to determine the IP address, where the port used is the default ManageSieve port of 2000.

2. Commands

This section and its subsections describes valid ManageSieve commands. Upon initial connection to the server the client's session is in non-authenticated state. Prior to successful authentication only the AUTHENTICATE, CAPABILITY, STARTTLS, LOGOUT and NOOP (see Section 2.13) commands are valid. ManageSieve extensions MAY define other commands which are valid in non-authenticated state. Servers MUST reject all other commands with a NO response. Clients may pipeline commands (send more than one command at a time without waiting for completion of the first command). However, a group of commands sent together MUST NOT have an AUTHENTICATE (*), a STARTTLS or a HAVESPACE command anywhere but the last command in the list.

(*) - The only exception to this rule is when the AUTHENTICATE command contains an initial response for a SASL mechanism that allows clients to send data first, the mechanism is known to complete in one round-trip and the mechanism doesn't negotiate a SASL security layer. Two examples of such SASL mechanisms are PLAIN [PLAIN] and EXTERNAL [SASL].

2.1. AUTHENTICATE Command

Arguments: String - mechanism
String - initial data (optional)

The AUTHENTICATE command indicates a SASL [SASL] authentication mechanism to the server. If the server supports the requested authentication mechanism, it performs an authentication protocol exchange to identify and authenticate the user. Optionally, it also negotiates a security layer for subsequent protocol interactions. If

the requested authentication mechanism is not supported, the server rejects the AUTHENTICATE command by sending the NO response.

The authentication protocol exchange consists of a series of server challenges and client responses that are specific to the selected authentication mechanism. A server challenge consists of a string (quoted or literal) followed by a CRLF. The contents of the string is a base-64 encoding [BASE64] of the SASL data. A client response consists of a string (quoted or literal) with the base-64 encoding of the SASL data followed by a CRLF. If the client wishes to cancel the authentication exchange, it issues a string containing a single "*". If the server receives such a response, it MUST reject the AUTHENTICATE command by sending an NO reply.

Note that an empty challenge/response is sent as an empty string. If the mechanism dictates that the final response is sent by the server this data MAY be placed within the data portion of the SASL response code to save a round trip.

The optional initial-response argument to the AUTHENTICATE command is used to save a round trip when using authentication mechanisms that are defined to send no data in the initial challenge. When the initial-response argument is used with such a mechanism, the initial empty challenge is not sent to the client and the server uses the data in the initial-response argument as if it were sent in response to the empty challenge. If the initial-response argument to the AUTHENTICATE command is used with a mechanism that sends data in the initial challenge, the server MUST reject the AUTHENTICATE command by sending the NO response.

The service name specified by this protocol's profile of SASL is "sieve".

Reauthentication is not supported by ManageSieve protocol's profile of SASL. I.e. after a successfully completed AUTHENTICATE command, no more AUTHENTICATE commands may be issued in the same session. After a successful AUTHENTICATE command completes, a server MUST reject any further AUTHENTICATE commands with a NO reply. However note that a server may implement UNAUTHENTICATE extension described in Section 2.14.1.

If a security layer is negotiated through the SASL authentication exchange, it takes effect immediately following the CRLF that concludes the successful authentication exchange for the client, and the CRLF of the OK response for the server.

When a security layer takes effect, the ManageSieve protocol is reset to the initial state (the state in ManageSieve after a client has

connected to the server). The server MUST discard any knowledge obtained from the client which was not obtained from the SASL (or TLS) negotiation itself. Likewise, the client MUST discard any knowledge obtained from the server, such as the list of ManageSieve extensions, which was not obtained from the SASL (and/or TLS) negotiation itself. (Note that a client MAY compare the advertised SASL mechanisms before and after authentication in order to detect an active down-negotiation attack. See below.)

Once a SASL security layer is established, the server MUST re-issue the capability results, followed by an OK response. This is necessary to protect against man-in-the-middle attacks which alter the capabilities list prior to SASL negotiation. The capability results MUST include all SASL mechanisms the server was capable of negotiating with that client. This is done in order to allow the client to detect active down-negotiation attack. If a user-oriented client detects such down-negotiation attack, it SHOULD either notify the user (it MAY give the user the opportunity to continue with the ManageSieve session in this case) or close the transport connection and indicate that a down-negotiation attack might be in progress. If an automated client detects down-negotiation attack, it SHOULD return or log an error indicating that a possible attack might be in progress and/or SHOULD close the transport connection.

When both [TLS] and SASL security layers are in effect, the TLS encoding MUST be applied (when sending data) after the SASL encoding.

Server implementations SHOULD support SASL proxy authentication so that an administrator can administer a user's scripts. Proxy authentication is when a user authenticates as herself/himself but requests the server to act (authorize) as another user.

The authorization identity generated by this [SASL] exchange is a "simple username" (in the sense defined in [SASLprep]), and both client and server MUST use the [SASLprep] profile of the [StringPrep] algorithm to prepare these names for transmission or comparison. If preparation of the authorization identity fails or results in an empty string (unless it was transmitted as the empty string), the server MUST fail the authentication.

If an AUTHENTICATE command fails with a NO response, the client MAY try another authentication mechanism by issuing another AUTHENTICATE command. In other words, the client may request authentication types in decreasing order of preference.

Note that a failed (NO) response to the AUTHENTICATE command may contain one of the following response codes: AUTH-TOO-WEAK, ENCRYPT-NEEDED or TRANSITION-NEEDED. See Section 1.4 for detailed

description of the relevant conditions.

To ensure interoperability, both client and server implementations of the ManageSieve protocol MUST implement the SCRAM-HMAC-SHA-1 [SCRAM] SASL mechanism, as well as [PLAIN] over [TLS].

Note: use of PLAIN over TLS reflects current use of PLAIN over TLS in other email related protocols, however a longer term goal is to migrate email related protocols from using PLAIN over TLS to SCRAM-HMAC-SHA-1 mechanism.

Examples (Note that long lines are folded for readability and are not part of protocol exchange):

```
S: "IMPLEMENTATION" "Example1 ManageSieved v001"
S: "SASL" "DIGEST-MD5 GSSAPI"
S: "SIEVE" "fileinto vacation"
S: "STARTTLS"
S: "VERSION" "1.0"
S: OK
C: Authenticate "DIGEST-MD5"
S: "cmVhbG09ImVsd29vZC5pbm5vc29mdC5leGFtcGxlLmNvbSIsbm9uY2U9Ik
   9BNk1HOXRFUUdtMmhoIixxb3A9ImF1dGgiLGFsZ29yaXRobT1tZDUtc2Vz
   cyxjaGFyc2V0PXV0Zi04"
C: "Y2hhcnNldD11dGYtOCx1c2VybmFtZT0iY2hyaXMiLHJlYWxtPSJlbHdvb2
   QuaW5ub3NvZnQuZXhhbXBsZS5jb20iLG5vbmNlPSJPQTZNRzl0RVFHbTJo
   aCIsbmM9MDAwMDAwMDEsY25vbmNlPSJPQTZNSFhoNlZxVHJSayIsZGlnZX
   N0LXVyaT0ic2lldmUvZWx3b29kLmlubm9zb2Z0LmV4YW1wbGUuY29tIixy
   ZXNwb25zZT1kMzg4ZGFkOTBkNGJiZDc2MGExNTIzMjFmMjE0M2FmNyxxb3
   A9YXV0aA=="
S: OK (SASL "cnNwYXV0aD1lYTQwZjYwMzM1YzQyN2I1NTI3Yjg0ZGJhYmNkZ
   mZmZA==")
```

A slightly different variant of the same authentication exchange:

```
S: "IMPLEMENTATION" "Example1 ManageSieved v001"
S: "SASL" "DIGEST-MD5 GSSAPI"
S: "SIEVE" "fileinto vacation"
S: "VERSION" "1.0"
S: "STARTTLS"
S: OK
C: Authenticate "DIGEST-MD5"
S: {136}
S: cmVhbG09ImVsd29vZC5pbm5vc29mdC5leGFtcGxlLmNvbSIsbm9uY2U9Ik
   9BNk1HOXRFUUdtMmhoIixxb3A9ImF1dGgiLGFsZ29yaXRobT1tZDUtc2Vz
   cyxjaGFyc2V0PXV0Zi04
C: {300+}
C: Y2hhcnNldD11dGYtOCx1c2VybmFtZT0iY2hyaXMiLHJlYWxtPSJlbHdvb2
   QuaW5ub3NvZnQuZXhhbXBsZS5jb20iLG5vbmNlPSJPQTZNRzl0RVFHbTJo
   aCIsbmM9MDAwMDAwMDEsY25vbmNlPSJPQTZNSFhoNlZxVHJSayIsZGlnZX
   N0LXVyaT0ic2lldmUvZWx3b29kLmlubm9zb2Z0LmV4YW1wbGUuY29tIixy
   ZXNwb25zZT1kMzg4ZGFkOTBkNGJiZDc2MGExNTIzMjFmMjE0M2FmNyxxb3
   A9YXV0aA==
S: {56}
S: cnNwYXV0aD1lYTQwZjYwMzM1YzQyN2I1NTI3Yjg0ZGJhYmNkZmZmZA==
C: ""
S: OK
```

Another example demonstrating use of SASL PLAIN mechanism under TLS. This example also demonstrate use of SASL "initial response" (the second parameter to the Authenticate command):

```
S: "IMPLEMENTATION" "Example1 ManageSieved v001"
S: "VERSION" "1.0"
S: "SASL" ""
S: "SIEVE" "fileinto vacation"
S: "STARTTLS"
S: OK
C: STARTTLS
S: OK
<TLS negotiation, further commands are under TLS layer>
S: "IMPLEMENTATION" "Example1 ManageSieved v001"
S: "VERSION" "1.0"
S: "SASL" "PLAIN"
S: "SIEVE" "fileinto vacation"
S: OK
C: Authenticate "PLAIN" "QJIrweAPyo6Q1T9xu"
S: NO
C: Authenticate "PLAIN" "QJIrweAPyo6Q1T9xz"
S: NO
C: Authenticate "PLAIN" "QJIrweAPyo6Q1T9xy"
S: BYE "Too many failed authentication attempts"
<Server closes connection>
```

The following example demonstrates use of SASL "initial response". It also demonstrates that an empty response can be sent as a literal, and that negotiation a SASL security layer results in the server reissuing server capabilities:

```
C: AUTHENTICATE "GSSAPI" {1488+}
C: YIIE[...1480 octets here ...]dA==
S: {208}
S: YIGZBgkqhkiG9xIBAgICAG+BiTCBhqADAgEFoQMCAQ+iejB4oAMCARKic
   [...114 octets here ...]
   /yzpAy9p+Y0LanLskOTvMc0MnjgAa4YEr3eJ6
C: {0+}
C:
S: {44}
S: BQQF/wAMAAwAAAAAYRGFAo6W0vIHti8i1UXODgEAEAA=
C: {44+}
C: BQQE/wAMAAwAAAAAIsT1iv9UkZApw471iXt6cwEAAAE=
S: OK
<Further commands/responses are under SASL security layer>
S: "IMPLEMENTATION" "Example1 ManageSieved v001"
S: "VERSION" "1.0"
S: "SASL" "PLAIN DIGEST-MD5 GSSAPI"
S: "SIEVE" "fileinto vacation"
S: "LANGUAGE" "ru"
S: "MAXREDIRECTS" "3"
S: ok
```

2.1.1. Use of SASL PLAIN mechanism over TLS

This section is normative for ManageSieve client implementations that support SASL [PLAIN] over [TLS].

If a ManageSieve client is willing to use SASL PLAIN over TLS to authenticate to the ManageSieve server, the client MUST verify the server identity (see Section 2.2.1). If the server identity can't be verified (e.g. the server has not provided any certificate, or if the certificate verification fails) the client MUST NOT attempt to authenticate using the SASL PLAIN mechanism.

2.2. STARTTLS Command

Support for STARTTLS command in servers is optional. Its availability is advertised with "STARTTLS" capability as described in Section 1.8.

The STARTTLS command requests commencement of a TLS [TLS] negotiation. The negotiation begins immediately after the CRLF in the OK response. After a client issues a STARTTLS command, it MUST

NOT issue further commands until a server response is seen and the TLS negotiation is complete.

The STARTTLS command is only valid in non-authenticated state. The server remains in non-authenticated state, even if client credentials are supplied during the TLS negotiation. The SASL [SASL] EXTERNAL mechanism MAY be used to authenticate once TLS client credentials are successfully exchanged, but servers supporting the STARTTLS command are not required to support the EXTERNAL mechanism.

After the TLS layer is established, the server MUST re-issue the capability results, followed by an OK response. This is necessary to protect against man-in-the-middle attacks which alter the capabilities list prior to STARTTLS. This capability result MUST NOT include the STARTTLS capability.

The client MUST discard cached capability information and replace it with the new information. The server MAY advertise different capabilities after STARTTLS.

Example:

```
C: StartTls
S: oK
<TLS negotiation, further commands are under TLS layer>
S: "IMPLEMENTATION" "Example1 ManageSieved v001"
S: "SASL" "PLAIN DIGEST-MD5 GSSAPI"
S: "SIEVE" "fileinto vacation"
S: "VERSION" "1.0"
S: "LANGUAGE" "fr"
S: ok
```

2.2.1. Server Identity Check

During the TLS negotiation, the ManageSieve client MUST check its understanding of the server hostname/IP address against the server's identity as presented in the server Certificate message, in order to prevent man-in-the-middle attacks. In this section, the client's understanding of the server's identity is called the "reference identity".

Checking is performed according to the following rules:

- If the reference identity is a hostname:

  1. If a subjectAltName extension of the SRVName [X509-SRV], dNSName [X509] (in that order of preference) type is present in the server's certificate, then it SHOULD be used as the

source of the server's identity. Matching is performed as described in Section 2.2.1.1, with the exception that no wildcard matching is allowed for SRVName type. If the certificate contains multiple names (e.g., more than one dNSName field), then a match with any one of the fields is considered acceptable.

2. The client MAY use other types of subjectAltName for performing comparison.

3. The server's identity MAY also be verified by comparing the reference identity to the Common Name (CN) [RFC4519] value in the leaf Relative Distinguished Name (RDN) of the subjectName field of the server's certificate. This comparison is performed using the rules for comparison of DNS names in Section 2.2.1.1, below, with the exception that no wildcard matching is allowed. [[anchor9: Chris Newman says that such prohibition of wildcards doesn't match existing practice.]] Although the use of the Common Name value is existing practice, it is deprecated, and Certification Authorities are encouraged to provide subjectAltName values instead. Note that the TLS implementation may represent DNs in certificates according to X.500 or other conventions. For example, some X.500 implementations order the RDNs in a DN using a left-to-right (most significant to least significant) convention instead of LDAP's right- to-left convention.

- When the reference identity is an IP address, the iPAddress subjectAltName SHOULD be used by the client for comparison. The comparison is performed as described in Section 2.2.1.2.

- In either case the client MAY map the reference identity to a different type prior to performing a comparison. Mappings may be performed for all available subjectAltName types to which the reference identity can be mapped; however, the reference identity should only be mapped to types for which the mapping is either inherently secure (e.g., extracting the DNS hostname from a URI) or for which the mapping is performed in a secure manner (e.g., using DNSSEC, or using user- or admin-configured host-to-address/address-to-host lookup tables).

If the server identity check fails, user-oriented clients SHOULD either notify the user (clients MAY give the user the opportunity to continue with the ManageSieve session in this case) or close the transport connection and indicate that the server's identity is suspect. Automated clients SHOULD return or log an error indicating that the server's identity is suspect and/or SHOULD close the transport connection. Automated clients MAY provide a configuration

setting that disables this check, but MUST provide a setting which enables it.

Beyond the server identity check described in this section, clients should be prepared to do further checking to ensure that the server is authorized to provide the service it is requested to provide. The client may need to make use of local policy information in making this determination.

2.2.1.1. Comparison of DNS Names

If the reference identity is an internationalized domain name, conforming implementations MUST convert it to the ASCII Compatible Encoding (ACE) format as specified in Section 4 of RFC 3490 [RFC3490] before comparison with subjectAltName values of type dNSName. Specifically, conforming implementations MUST perform the conversion operation specified in Section 4 of [RFC3490] as follows:

- in step 1, the domain name SHALL be considered a "stored string";
- in step 3, set the flag called "UseSTD3ASCIIRules";
- in step 4, process each label with the "ToASCII" operation; and
- in step 5, change all label separators to U+002E (full stop).

After performing the "to-ASCII" conversion, the DNS labels and names MUST be compared for equality according to the rules specified in Section 3 of [RFC3490], i.e. once all label separators are replaced with U+002E (dot) they are compared in the case-insensitive manner.

The '*' (ASCII 42) wildcard character is allowed in subjectAltName values of type dNSName, and then only as the left-most (least significant) DNS label in that value. This wildcard matches any left-most DNS label in the server name. That is, the subject *.example.com matches the server names a.example.com and b.example.com, but does not match example.com or a.b.example.com.

2.2.1.2. Comparison of IP Addresses

When the reference identity is an IP address, the identity MUST be converted to the "network byte order" octet string representation [RFC791][RFC2460]. For IP Version 4, as specified in RFC 791, the octet string will contain exactly four octets. For IP Version 6, as specified in RFC 2460, the octet string will contain exactly sixteen octets. This octet string is then compared against subjectAltName values of type iPAddress. A match occurs if the reference identity octet string and value octet strings are identical.

2.2.1.3. Comparison of Other subjectName Types

Client implementations MAY support matching against subjectAltName values of other types as described in other documents.

2.3. LOGOUT Command

The client sends the LOGOUT command when it is finished with a connection and wishes to terminate it. The server MUST reply with an OK response and terminate the connection. The server MUST ignore commands issued by the client after the LOGOUT command.

Example:

```
C: Logout
S: Ok
<connection terminated>
```

2.4. CAPABILITY Command

The CAPABILITY command requests the server capabilities as described earlier in this document. It has no parameters.

Example:

```
C: CAPABILITY
S: "IMPLEMENTATION" "Example1 ManageSieved v001"
S: "VERSION" "1.0"
S: "SASL" "PLAIN OTP GSSAPI"
S: "SIEVE" "fileinto vacation"
S: "STARTTLS"
S: OK
```

2.5. HAVESPACE Command

```
Arguments:  String - name
            Number - script size
```

The HAVESPACE command is used to query the server for available space. Clients specify the name they wish to save the script as and its size in octets. Both parameters can be used by the server to see if the script with the specified name and size is within user's quota(s), for example the server MAY use the script name to check if a script would be replaced or a new one would be created. Servers respond with an NO if storing a script with that name and size would fail or OK otherwise. Clients SHOULD issue this command before attempting to place a script on the server.

Note that the OK response from the HAVESPACE command does not constitute a guarantee of success as server disk space conditions could change between the client issuing the HAVESPACE and the client issuing the PUTSCRIPT commands. A QUOTA response code (see Section 1.4) remains a possible (albeit unlikely) response to a subsequent PUTSCRIPT with the same name and size.

Example:

```
C: HAVESPACE "myscript" 999999
S: NO (QUOTA/MAXSIZE) "Quota exceeded"

C: HAVESPACE "foobar" 435
S: OK
```

2.6. PUTSCRIPT Command

Arguments: String - Script name
String - Script content

The PUTSCRIPT command is used by the client to submit a Sieve script to the server.

If the script already exists, upon success the old script will be overwritten. The old script MUST NOT be overwritten if PUTSCRIPT fails in any way. A script of zero length SHOULD be disallowed.

This command places the script on the server. It does not affect whether the script is processed on incoming mail, unless it replaces the script which is already active. The SETACTIVE command is used to mark a script as active.

When submitting large scripts clients SHOULD use the HAVESPACE command beforehand to query if the server is willing to accept a script of that size.

The server MUST check the submitted script for validity, which includes checking that the script complies with the Sieve grammar [SIEVE], and that all Sieve extensions mentioned in script's "require" statement(s) are supported by the Sieve interpreter. (Note that if the Sieve interpreter supports the Sieve "ihave" extension [I-HAVE], any unrecognized/unsupported extension mentioned in the "ihave" test MUST NOT cause the validation failure.) Other checks such as validating the supplied command arguments for each command MAY be performed. Essentially, the performed validation SHOULD be the same as performed when compiling the script for execution. Implementations that use a binary representation to store compiled scripts can extend the validation to a full compilation, in order to

avoid validating uploaded scripts multiple times.

If the script fails the validation the server MUST reply with a NO response. Any script that fails the validity test MUST NOT be stored on the server. The message given with a NO response MUST be human readable and SHOULD contain a specific error message giving the line number of the first error. Implementors should strive to produce helpful error messages similar to those given by programming language compilers. Client implementations should note that this may be a multiline literal string with more than one error message separated by CRLFs. The human readable message is in the language returned in the latest LANGUAGE capability (or in "i-default", see Section 1.8), encoded in UTF-8 [UTF-8].

An OK response MAY contain the WARNINGS response code. In such case the human readable message that follows the OK response SHOULD contain a specific warning message (or messages) giving the line number(s) in the script that might contain errors not intended by the script writer. The human readable message is in the language returned in the latest LANGUAGE capability (or in "i-default", see Section 1.8), encoded in UTF-8 [UTF-8] A client seeing such response code SHOULD present the message to the user.

Examples:

```
C: Putscript "foo" {31+}
C: #comment
C: InvalidSieveCommand
C:
S: NO "line 2: Syntax error"

C: Putscript "mysievescript" {110+}
C: require ["fileinto"];
C:
C: if envelope :contains "to" "tmartin+sent" {
C:   fileinto "INBOX.sent";
C: }
S: OK

C: Putscript "myforwards" {190+}
C: redirect "111@example.net";
C:
C: if size :under 10k {
C:     redirect "mobile@cell.example.com";
C: }
C:
C: if envelope :contains "to" "tmartin+lists" {
C:     redirect "lists@groups.example.com";
C: }
S: OK (WARNINGS) "line 8: server redirect action
        limit is 2, this redirect might be ignored"
```

2.7. LISTSCRIPTS Command

This command lists the scripts the user has on the server. Upon success a list of CRLF separated script names (each represented as a quoted or literal string) is returned followed by an OK response. If there exists an active script the atom ACTIVE is appended to the corresponding script name. The atom ACTIVE MUST NOT appear on more than one response line.

Example:

```
C: Listscripts
S: "summer_script"
S: "vacation_script"
S: {13}
S: clever"script
S: "main_script" ACTIVE
S: OK

C: listscripts
S: "summer_script"
S: "main_script" active
S: OK
```

2.8. SETACTIVE Command

Arguments: String - script name

This command sets a script active. If the script name is the empty string (i.e. "") then any active script is disabled. Disabling an active script when there is no script active is not an error and MUST result in OK reply.

If the script does not exist on the server then the server MUST reply with a NO response. Such reply SHOULD contain the NONEXISTENT response code.

Examples:

```
C: Setactive "vacationscript"
S: Ok

C: Setactive ""
S: Ok

C: Setactive "baz"
S: No (NONEXISTENT) "There is no script by that name"

C: Setactive "baz"
S: No (NONEXISTENT) {31}
S: There is no script by that name
```

2.9. GETSCRIPT Command

Arguments: String - script name

This command gets the contents of the specified script. If the script does not exist the server MUST reply with a NO response. Such reply SHOULD contain the NONEXISTENT response code.

Upon success a string with the contents of the script is returned followed by a OK response.

Example:

```
C: Getscript "myscript"
S: {54}
S: #this is my wonderful script
S: reject "I reject all";
S:
S: OK
```

2.10. DELETESCRIPT Command

Arguments: String - script name

This command is used to delete a user's Sieve script. Servers MUST reply with a NO response if the script does not exist. Such responses SHOULD include the NONEXISTENT response code.

The server MUST NOT allow the client to delete an active script, so the server MUST reply with a NO response if attempted. Such response SHOULD contain the ACTIVE response code. If a client wishes to delete an active script it should use the SETACTIVE command to disable the script first.

Example:

```
C: Deletescript "foo"
S: Ok

C: Deletescript "baz"
S: No (ACTIVE) "You may not delete an active script"
```

2.11. RENAMESCRIPT Command

Arguments: String - Old Script name
String - New Script name

This command is used to rename a user's Sieve script. Servers MUST reply with a NO response if the old script does not exist (in which case the NONEXISTENT response code SHOULD be included), or a script

with the new name already exists (in which case the ALREADYEXISTS response code SHOULD be included). Renaming the active script is allowed, the renamed script remains active.

Example:

```
C: Renamescript "foo" "bar"
S: Ok

C: Renamescript "baz" "bar"
S: No "bar already exists"
```

If the server doesn't support the RENAMESCRIPT command, the client can emulate it by performing the following steps:

1. List available scripts with LISTSCRIPTS. If the script with the new script name exists, then the client should ask the user whether to abort the operation, to replace the script (by issuing the DELETESCRIPT <newname> after that) or to chose a different name.

2. Download the old script with GETSCRIPT <oldname>.

3. Upload the old script with the new name: PUTSCRIPT <newname>.

4. If the old script was active (as reported by LISTSCRIPTS in step 1), then make the new script active: SETACTIVE <newname>

5. Delete the old script: DELETESCRIPT <oldname>

Note that these steps don't describe how to handle various other error conditions (for example NO response containing QUOTA response code in step 3). Error handling is left as an excercise for the reader.

2.12. CHECKSCRIPT Command

Arguments: String - Script content

The CHECKSCRIPT command is used by the client to verify Sieve script validity without storing the script on the server.

The server MUST check the submitted script for syntactic validity, which includes checking that all Sieve extensions mentioned in Sieve script "require" statement(s) are supported by the Sieve interpreter. (Note that if the Sieve interpreter supports the Sieve "ihave" extension [I-HAVE], any unrecognized/unsupported extension mentioned in the "ihave" test MUST NOT cause the syntactic validation failure.)

If the script fails this test the server MUST reply with a NO response. The message given with a NO response MUST be human readable and SHOULD contain a specific error message giving the line number of the first error. Implementors should strive to produce helpful error messages similar to those given by programming language compilers. Client implementations should note that this may be a multiline literal string with more than one error message separated by CRLFs. The human readable message is in the language returned in the latest LANGUAGE capability (or in "i-default", see Section 1.8), encoded in UTF-8 [UTF-8].

Examples:

```
C: CheckScript {31+}
C: #comment
C: InvalidSieveCommand
C:
S: NO "line 2: Syntax error"
```

A ManageSieve server supporting this command MUST NOT check if the script will put the current user over its quota limit.

An OK response MAY contain the WARNINGS response code. In such case the human readable message that follows the OK response SHOULD contain a specific warning message (or messages) giving the line number(s) in the script that might contain errors not intended by the script writer. The human readable message is in the language returned in the latest LANGUAGE capability (or in "i-default", see Section 1.8), encoded in UTF-8 [UTF-8] A client seeing such response code SHOULD present the message to the user.

2.13. NOOP Command

Arguments: String - tag to echo back (optional)

The NOOP command does nothing, beyond returning a response to the client. It may be used by clients for protocol re-synchronisation or to reset any inactivity auto-logout timer on the server.

The response to the NOOP command is always OK, followed by the TAG response code together with the supplied string; if no string was supplied in the NOOP command, the TAG response code MUST NOT be included.

Examples:

```
C: NOOP
S: OK "NOOP completed"

C: NOOP "STARTTLS-SYNC-42"
S: OK (TAG {16}
S: STARTTLS-SYNC-42) "Done"
```

2.14. Recommended extensions

The UNAUTHENTICATE extension (advertised as the "UNAUTHENTICATE" capability with no parameters) defines a new UNAUTHENTICATE command, which allows a client to return the server to non-authenticated state. Support for this extension is RECOMMENDED.

2.14.1. UNAUTHENTICATE Command

The UNAUTHENTICATE command returns the server to the non-authenticated state. It doesn't affect any previously established TLS [TLS] or SASL (Section 2.1) security layer.

The UNAUTHENTICATE command is only valid in authenticated state. If issued in a wrong state, the server MUST reject it with a NO response.

The UNAUTHENTICATE command has no parameters.

When issued in the authenticated state, the UNAUTHENTICATE command MUST NOT fail (i.e. it must never return anything other than OK or BYE)

3. Sieve URL Scheme

URI scheme name: sieve

Status: permanent

URI scheme syntax:

Described using ABNF [ABNF]. Some ABNF productions not defined below are from [URI-GEN].

```
sieveurl = sieveurl-server / sieveurl-list-scripts /
           sieveurl-script

sieveurl-server = "sieve://" authority

sieveurl-list-scripts = "sieve://" authority ["/"]

sieveurl-script = "sieve://" authority "/"
                  [owner "/"] scriptname

authority = <defined in [URI-GEN]>

owner         = *ochar
                ;; %-encoded version of [SASL] authorization
                ;; identity (script owner) or "userid".
                ;;
                ;; Empty owner is used to reference
                ;; global scripts.
                ;;
                ;; Note that ASCII characters such as " ", ";",
                ;; "&", "=", "/" and "?" must be %-encoded
                ;; as per rule specified in [URI-GEN].

scriptname    = 1*ochar
                ;; %-encoded version of UTF-8 representation
                ;; of the script name.
                ;; Note that ASCII characters such as " ", ";",
                ;; "&", "=", "/" and "?" must be %-encoded
                ;; as per rule specified in [URI-GEN].

ochar         = unreserved / pct-encoded / sub-delims-sh /
                ":" / "@"
                ;; Same as [URI-GEN] 'pchar'
                ;; but without ";", "&" and "=".

unreserved = <defined in [URI-GEN]>

pct-encoded = <defined in [URI-GEN]>

sub-delims-sh = "!" / "$" / "'" / "(" / ")" /
                "*" / "+" / ","
                ;; Same as [URI-GEN] sub-delims,
                ;; but without ";", "&" and "=".
```

URI scheme semantics:

A Sieve URL identifies a Sieve server or a Sieve script on a Sieve server. The latter form is associated with the application/sieve MIME type defined in [SIEVE]. There is no MIME type associated with the former form of Sieve URI.

The server form is used in the REFERRAL response code (see Section 1.4 in order to designate another server where the client should perform its operations.

The script form allows to retrieve (GETSCRIPT), update (PUTSCRIPT), delete (DELETESCRIPT) or activate (SETACTIVE) the named script, however the most typical action would be to retrieve the script. If the script name is empty (omitted), the URI requests that the client lists available scripts using the LISTSCRIPTS command.

Encoding considerations:

The script name and/or the owner, if present, is in UTF-8. Non-US-ASCII UTF-8 octets MUST be percent-encoded as described in [URI-GEN]. US-ASCII characters such as " " (space), ";", "&", "=", "/" and "?" MUST be %-encoded as described in [URI-GEN]. Note that "&" and "?" are in this list in order to allow for future extensions.

Note that the empty owner (e.g. sieve://example.com//script) is different from the missing owner (e.g. sieve://example.com/script) and is reserved for referencing global scripts.

The user name (in the "authority" part), if present, is in UTF-8. Non-US-ASCII UTF-8 octets MUST be percent-encoded as described in [URI-GEN].

Applications/protocols that use this URI scheme name:
ManageSieve [RFC XXXX] clients and servers. Clients that can store user preferences in protocols such as [LDAP] or [ACAP].

Interoperability considerations: None.

Security considerations:
The <scriptname> part of a ManageSieve URL might potentially disclose some confidential information about the author of the script or, depending on a ManageSieve implementation, about configuration of the mail system. The latter might be used to prepare for a more complex attack on the mail system.

Clients resolving ManageSieve URLs that wish to achieve data confidentiality and/or integrity SHOULD use the STARTTLS command (if supported by the server) before starting authentication, or use a SASL mechanism, such as GSSAPI, that provides a confidentiality security layer.

Contact: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

References: This document and RFC 5228 [SIEVE].

4. Formal Syntax

The following syntax specification uses the augmented Backus-Naur Form (BNF) notation as specified in [ABNF]. This uses the ABNF core rules as specified in Appendix A of the ABNF specification [ABNF]. "UTF8-2", "UTF8-3" and "UTF8-4" non-terminal are defined in [UTF-8].

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.

```
SAFE-CHAR             = %x01-09 / %x0B-0C / %x0E-21 / %x23-5B /
                        %x5D-7F
                        ;; any TEXT-CHAR except QUOTED-SPECIALS

QUOTED-CHAR           = SAFE-UTF8-CHAR / DQUOTE QUOTED-SPECIALS

QUOTED-SPECIALS       = DQUOTE / "\"

SAFE-UTF8-CHAR        = SAFE-CHAR / UTF8-2 / UTF8-3 / UTF8-4
                        ;; <UTF8-2>, <UTF8-3> and <UTF8-4>
                        ;; are defined in [UTF-8]

ATOM-CHAR             = "!" / %x23-27 / %x2A-5B / %x5D-7A / %x7C-7E
                        ;; Any CHAR except ATOM-SPECIALS

ATOM-SPECIALS         = "(" / ")" / "{" / SP / CTL /
                         QUOTED-SPECIALS

atom                  = 1*1024ATOM-CHAR

iana-token            = atom
                        ;; MUST be registered with IANA
```

```
    auth-type             = DQUOTE auth-type-name DQUOTE

    auth-type-name        = iana-token
                            ;; as defined in SASL [SASL]

    command               = (command-any / command-auth /
                             command-nonauth) CRLF
                            ;; Modal based on state

    command-any           = command-capability / command-logout /
                            command-noop
                            ;; Valid in all states

    command-auth          = command-getscript / command-setactive /
                            command-listscripts / command-deletescript /
                            command-putscript / command-checkscript /
                            command-havespace /  /
                            command-renamescript /
                            command-unauthenticate
                            ;; Valid only in Authenticated state

    command-nonauth       = command-authenticate / command-starttls
                            ;; Valid only when in Non-Authenticated
                            ;; state

    command-authenticate  = "AUTHENTICATE" SP auth-type [SP string]
                            *(CRLF string)

    command-capability    = "CAPABILITY"

    command-deletescript  = "DELETESCRIPT" SP sieve-name

    command-getscript     = "GETSCRIPT" SP sieve-name

    command-havespace     = "HAVESPACE" SP sieve-name SP number

    command-listscripts   = "LISTSCRIPTS"

    command-noop          = "NOOP" [SP string]

    command-logout        = "LOGOUT"

    command-putscript     = "PUTSCRIPT" SP sieve-name SP sieve-script

    command-checkscript   = "CHECKSCRIPT" SP sieve-script

    sieve-script          = string
```

```
    command-renamescript  = "RENAMESCRIPT" SP old-sieve-name SP
                            new-sieve-name

    old-sieve-name        = sieve-name

    new-sieve-name        = sieve-name

    command-setactive     = "SETACTIVE" SP active-sieve-name

    command-starttls      = "STARTTLS"

    command-unauthenticate= "UNAUTHENTICATE"

    extend-token          = atom
                            ;; MUST be defined by a standards track or
                            ;; IESG approved experimental protocol
                            ;; extension

    extension-data        = extension-item *(SP extension-item)

    extension-item        = extend-token / string / number /
                            "(" [extension-data] ")"

    literal-c2s           = "{" number "+}" CRLF *OCTET
                            ;; The number represents the number of
                            ;; octets.
                            ;; This type of literal can only be sent
                            ;; from the client to the server.

    literal-s2c           = "{" number "}" CRLF *OCTET
                            ;; Almost identical to literal-c2s,
                            ;; but with no '+' character.
                            ;; The number represents the number of
                            ;; octets.
                            ;; This type of literal can only be sent
                            ;; from the server to the client.

    number                = 1*DIGIT
                            ;; A 32-bit unsigned number.
                            ;; (0 <= n < 4,294,967,296)

    number-str            = string
                            ;; <number> encoded as a <string>.

    quoted                = DQUOTE *1024QUOTED-CHAR DQUOTE
                            ;; limited to 1024 octets between the <">s

    resp-code             = "AUTH-TOO-WEAK" / "ENCRYPT-NEEDED" /
```

```
                          "QUOTA" ["/" ("MAXSCRIPTS" / "MAXSIZE")] /
                          resp-code-sasl /
                          resp-code-referral /
                          "TRANSITION-NEEDED" / "TRYLATER" /
                          "ACTIVE" / "NONEXISTENT" /
                          "ALREADYEXISTS" / "WARNINGS" /
                          "TAG" SP string /
                          resp-code-ext

    resp-code-referral    = "REFERRAL" SP sieveurl

    resp-code-sasl        = "SASL" SP string

    resp-code-name        = iana-token
                          ;; The response code name is hierarchical,
                          ;; separated by '/'.
                          ;; The response code name MUST NOT start
                          ;; with '/'.

    resp-code-ext         = resp-code-name [SP extension-data]
                          ;; unknown response codes MUST be tolerated
                          ;; by the client.

    response              = response-authenticate /
                          response-logout /
                          response-getscript /
                          response-setactive /
                          response-listscripts /
                          response-deletescript /
                          response-putscript /
                          response-checkscript /
                          response-capability /
                          response-havespace /
                          response-starttls /
                          response-renamescript /
                          response-noop /
                          response-unauthenticate

    response-authenticate = *(string CRLF)
                          ((response-ok [response-capability]) /
                           response-nobye)
                          ;; <response-capability> is REQUIRED if a
                          ;; SASL security layer was negotiated and
                          ;; MUST be omitted otherwise.

    response-capability   = *(single-capability) response-oknobye

    single-capability     = capability-name [SP string] CRLF
```

```
    capability-name       = string
                            ;; Note that literal-s2c is allowed.

    initial-capabilities  = DQUOTE "IMPLEMENTATION" DQUOTE SP string /
                            DQUOTE "SASL" DQUOTE SP sasl-mechs /
                            DQUOTE "SIEVE" DQUOTE SP sieve-extensions /
                            DQUOTE "MAXREDIRECTS" DQUOTE SP number-str /
                            DQUOTE "NOTIFY" DQUOTE SP notify-mechs /
                            DQUOTE "STARTTLS" DQUOTE /
                            DQUOTE "LANGUAGE" DQUOTE SP language /
                            DQUOTE "VERSION" DQUOTE SP version /
                            DQUOTE "OWNER" DQUOTE SP string
                            ;; Each capability conforms to
                            ;; the syntax for single-capability.
                            ;; Also note that the capability name
                            ;; can be returned as either literal-s2c
                            ;; or quoted, even though only "quoted"
                            ;; string is shown above.
    version = DQUOTE "1.0" DQUOTE

    sasl-mechs = string
                 ; space separated list of SASL mechanisms,
                 ; each SASL mechanism name complies with rules
                 ; specified in [SASL].
                 ; Can be empty.

    sieve-extensions = string
                 ; space separated list of supported SIEVE extensions,
                 ; can be empty.

    language     = string
                 ; Contains <Language-Tag> from [RFC4646].

    notify-mechs = string
                 ; space separated list of URI schema parts
                 ; for supported notification [NOTIFY] methods.
                 ; MUST NOT be empty.

    response-deletescript = response-oknobye

    response-getscript    = (sieve-script CRLF response-ok) /
                            response-nobye

    response-havespace    = response-oknobye

    response-listscripts  = *(sieve-name [SP "ACTIVE"] CRLF)
                            response-oknobye
                            ;; ACTIVE may only occur with one sieve-name
```

```
    response-logout       = response-oknobye

    response-unauthenticate= response-oknobye
                             ;; "NO" response can only be returned when
                             ;; the command is issued in a wrong state
                             ;; or has a wrong number of parameters

    response-ok           = "OK" [SP "(" resp-code ")"]
                            [SP string] CRLF
                            ;; The string contains human readable text
                            ;; encoded as UTF-8.

    response-nobye        = ("NO" / "BYE") [SP "(" resp-code ")"]
                            [SP string] CRLF
                            ;; The string contains human readable text
                            ;; encoded as UTF-8.

    response-oknobye      = response-ok / response-nobye

    response-noop         = response-ok

    response-putscript    = response-oknobye

    response-checkscript  = response-oknobye

    response-renamescript = response-oknobye

    response-setactive    = response-oknobye

    response-starttls     = (response-ok response-capability) /
                            response-nobye

    sieve-name            = string
                            ;; See Section 1.6 for the full list of
                            ;; prohibited characters.
                            ;; Empty string is not allowed.

    active-sieve-name     = string
                            ;; See Section 1.6 for the full list of
                            ;; prohibited characters.
                            ;; This is similar to <sieve-name>, but
                            ;; empty string is allowed and has a special
                            ;; meaning.

    string                = quoted / literal-c2s / literal-s2c
                            ;; literal-c2s is only allowed when sent
                            ;; from the client to the server.
                            ;; literal-s2c is only allowed when sent
```

```
                    ;; from the server to the client.
                    ;; quoted is allowed in either direction.
```

5. Security Considerations

The AUTHENTICATE command uses SASL [SASL] to provide authentication and authorization services. Integrity and privacy services can be provided by [SASL] and/or [TLS]. When a SASL mechanism is used the security considerations for that mechanism apply.

This protocol's transactions are susceptible to passive observers or man in the middle attacks which alter the data, unless the optional encryption and integrity services of the SASL (via the AUTHENTICATE command) and/or [TLS] (via the STARTTLS command) are enabled, or an external security mechanism is used for protection. It may be useful to allow configuration of both clients and servers to refuse to transfer sensitive information in the absence of strong encryption.

If an implementation supports SASL mechanisms that are vulnerable to passive eavesdropping attacks (such as [PLAIN]), then the implementation MUST support at least one configuration where these SASL mechanisms are not advertised or used without the presence of an external security layer such as [TLS].

Some response codes returned on failed AUTHENTICATE command may disclose whether or not the username is valid, so server implementations SHOULD provide the ability to disable these features (or make them not conditional on a per-user basis) for sites concerned about such disclosure. In the case of ENCRYPT-NEEDED, if it is applied to all identities then no extra information is disclosed, but if it is applied on a per-user basis it can disclose information.

6. IANA Considerations

IANA is requested to reserve TCP port number 2000 for use with the ManageSieve protocol described in this document.

IANA is requested to register the "sieve" URI scheme defined in Section 3 of this document.

IANA is requested to create a new registry for ManageSieve capabilities. The registration template for ManageSieve capabilities is specified in Section 6.1. ManageSieve protocol capabilities MUST be specified in a standards track or IESG approved experimental RFC.

IANA is requested to create a new registry for ManageSieve response codes. The registration template for ManageSieve response codes is specified in Section 6.3. ManageSieve protocol response codes MUST be specified in a standards track or IESG approved experimental RFC.

6.1. ManageSieve Capability Registration Template

To: iana@iana.org
Subject: ManageSieve Capability Registration

Please register the following ManageSieve Capability:
Capability name:
Description:
Relevant publications:
Person & email address to contact for further information:
Author/Change controller:

6.2. Registration of Initial ManageSieve capabilities

To: iana@iana.org
Subject: ManageSieve Capability Registration

Please register the following ManageSieve Capabilities:

Capability name: IMPLEMENTATION

Description: Its value contains name of server implementation and its version.

Relevant publications: this RFC, Section 1.8.

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Capability name: SASL

Description: Its value contains a space separated list of SASL mechanisms supported by server.

Relevant publications: this RFC, Section 1.8 and Section 2.1.

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Capability name: SIEVE

Description: Its value contains a space separated list of supported SIEVE extensions

Relevant publications: this RFC, Section 1.8. Also [SIEVE].

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Capability name: STARTTLS

Description: This capability is returned if server supports TLS (STARTTLS command).

Relevant publications: this RFC, Section 1.8 and Section 2.2.

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Capability name: NOTIFY

Description: This capability is returned if server supports 'enotify' [NOTIFY] Sieve extension.

Relevant publications: this RFC, Section 1.8.

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Capability name: OWNER

Description: Its value contains UTF-8 encoded name of the currently logged in user ("authorization identity" according to RFC 4422).

Relevant publications: this RFC, Section 1.8.

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Capability name: VERSION

Description: This capability is returned if the server is compliant with RFCXXXX, i.e. that it supports RENAMESCRIPT, CHECKSCRIPT and NOOP commands.

Relevant publications: this RFC, Section 2.11.

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

6.3. ManageSieve Response Code Registration Template

To: iana@iana.org
Subject: ManageSieve Response Code Registration

Please register the following ManageSieve Response Code:

Response Code:

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified):

Purpose:

Published Specification(s):

Person & email address to contact for further information:

Author/Change controller:

6.4. Registration of Initial ManageSieve Response Codes

To: iana@iana.org
Subject: ManageSieve Response Code Registration

Please register the following ManageSieve Response Codes:

Response Code: AUTH-TOO-WEAK

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified): NONE

Purpose: This response code is returned in the NO response from an AUTHENTICATE command. It indicates that site security policy forbids the use of the requested mechanism for the specified authentication identity.

Published Specification(s): [RFCXXXX]

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Response Code: ENCRYPT-NEEDED

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified): NONE

Purpose: This response code is returned in the NO response from an AUTHENTICATE command. It indicates that site security policy requires the use of a strong encryption mechanism for the specified authentication identity and mechanism.

Published Specification(s): [RFCXXXX]

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Response Code: QUOTA

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified): NONE

Purpose: If this response code is returned in the NO/BYE response, it means that the command would have placed the user above the site-defined quota constraints. If this response code is returned in the OK response, it can mean that the user is near its quota or that the user exceeded its quota, but the server supports soft quotas.

Published Specification(s): [RFCXXXX]

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Response Code: QUOTA/MAXSCRIPTS

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified): NONE

Purpose: If this response code is returned in the NO/BYE response, it means that the command would have placed the user above the site-defined limit on the number of Sieve scripts. If this response code is returned in the OK response, it can mean that the user is near its quota or that the user exceeded its quota, but the server supports soft quotas. This response code is a more specific version of the QUOTA response code.

Published Specification(s): [RFCXXXX]

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Response Code: QUOTA/MAXSIZE

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified): NONE

Purpose: If this response code is returned in the NO/BYE response, it means that the command would have placed the user above the site-defined maximum script size. If this response code is returned in the OK response, it can mean that the user is near its quota or that the user exceeded its quota, but the server supports soft quotas. This response code is a more specific version of the QUOTA response code.

Published Specification(s): [RFCXXXX]

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Response Code: REFERRAL

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified): <sieveurl>

Purpose: This response code may be returned with a BYE result from any command, and includes a mandatory parameter that indicates what server to access to manage this user's sieve scripts. The

server will be specified by a Sieve URL (see Section 3). The scriptname portion of the URL MUST NOT be specified. The client should authenticate to the specified server and use it for all further commands in the current session.

Published Specification(s): [RFCXXXX]

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Response Code: SASL

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified): <string>

Purpose: This response code can occur in the OK response to a successful AUTHENTICATE command and includes the optional final server response data from the server as specified by [SASL].

Published Specification(s): [RFCXXXX]

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Response Code: TRANSITION-NEEDED

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified): NONE

Purpose: This response code occurs in a NO response of an AUTHENTICATE command. It indicates that the user name is valid, but the entry in the authentication database needs to be updated in order to permit authentication with the specified mechanism. This is typically done by establishing a secure channel using TLS, followed by authenticating once using the [PLAIN] authentication mechanism. The selected mechanism SHOULD then work for authentications in subsequent sessions.

Published Specification(s): [RFCXXXX]

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Response Code: TRYLATER

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified): NONE

Purpose: A command failed due to a temporary server failure. The client MAY continue using local information and try the command later. This response code only make sense when returned in a NO/BYE response.

Published Specification(s): [RFCXXXX]

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Response Code: ACTIVE

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified): NONE

Purpose: A command failed because it is not allowed on the active script. For example DELETESCRIPT on the active script. This response code only makes sense when returned in a NO/BYE response.

Published Specification(s): [RFCXXXX]

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Response Code: NONEXISTENT

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified): NONE

Purpose: A command failed because the referenced script name doesn't exist. This response code only makes sense when returned in a NO/BYE response.

Published Specification(s): [RFCXXXX]

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Response Code: ALREADYEXISTS

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified): NONE

Purpose: A command failed because the referenced script name already exists. This response code only makes sense when returned in a NO/BYE response.

Published Specification(s): [RFCXXXX]

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Response Code: WARNINGS

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified): NONE

Purpose: This response code MAY be returned by the server in the OK response (but it might be returned with the NO/BYE response as well) and signals the client that even though the script is syntactically valid, it might contain errors not intended by the script writer.

Published Specification(s): [RFCXXXX]

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

Response Code: TAG

Arguments (use ABNF to specify syntax, or the word NONE if none can be specified): string

Purpose: This response code name is followed by a string specified in the command that caused this response. It is typically used for client state synchronization.

Published Specification(s): [RFCXXXX]

Person & email address to contact for further information: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG.

7. Internationalization Considerations

The LANGUAGE capability (see Section 1.8) allows a client to discover the current language used in all human readable responses that might be returned at the end of any OK/NO/BYE response. Human readable text in OK responses typically doesn't need to be shown to the user, unless it is returned in response to PUTSCRIPT or CHECKSCRIPT command that also contain the WARNINGS response code Section 1.4. Human readable text from NO/BYE responses is intended be shown to the user, unless the client can automatically handle failure of the command that caused such response. Clients SHOULD use response codes (Section 1.4) for automatic error handling. Response codes MAY also be used by the client to present error messages in a language understood by the user, for example if the LANGUAGE capability doesn't return a language understood by the user.

Note that the human readable text from OK (WARNINGS) or NO/BYE responses for PUTSCRIPT/CHECKSCRIPT commands is intended for advanced users that understand Sieve language. Such advanced users are often sophisticated enough to be able to handle whatever language the server is using, even if it is not their preferred language, and will want to see error/warning text no matter what language the server puts it in.

A client that generates Sieve script automatically, for example if the script is generated without user intervention or from a UI that presents an abstract list of conditions and corresponding actions, SHOULD NOT present warning/error messages to the user, because the user might not even be aware that the client is using Sieve underneath. However if the client has a debugging mode, such warnings/errors SHOULD be available in the debugging mode.

8. Acknowledgements

Thanks to Simon Josefsson, Larry Greenfield, Allen Johnson, Chris Newman, Lyndon Nerenberg, Tim Showalter, Sarah Robeson, Walter Wong, Barry Leiba, Arnt Gulbrandsen, Stephan Bosch, Ken Murchison, Phil Pennock, Ned Freed, Jeffrey Hutzelman, Mark E. Mallett, Dilyan Palauzov, Dave Cridland, Aaron Stone, Robert Burrell Donkin, Patrick Ben Koetter, Bjoern Hoehrmann and Martin Duerst for help with this document. Special thank you to Phil Pennock for providing text for

the NOOP command, as well as finding various bugs in the document.

9. References

9.1. Normative References

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

[ACAP] Newman, C. and J. Myers, "ACAP -- Application Configuration Access Protocol", RFC 2244, November 1997.

[BASE64] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

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

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

[NET-UNICODE] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, March 2008.

[NOTIFY] Melnikov, A., Ed., Leiba, B., Ed., Segmuller, W., and T. Martin, "Sieve Extension: Notifications", draft-ietf-sieve-notify-12 (work in progress), December 2007.

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", RFC 2277, January 1998.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[RFC4519] Sciberras, A., "Lightweight Directory Access Protocol (LDAP): Schema for User Applications", RFC 4519, June 2006.

[RFC4646] Phillips, A. and M. Davis, "Tags for Identifying

Languages", RFC 4646, September 2006.

[RFC791] Postel, J., "Internet Protocol", RFC 791, September 1981.

[SASL] Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.

[SASLprep] Zeilenga, K., "SASLprep: Stringprep Profile for User Names and Passwords", RFC 4013, February 2005.

[SCRAM] Menon-Sen, A., Ed. and C. Newman, "Salted Challenge Response Authentication Mechanism (SCRAM)", draft-newman-auth-scram-07.txt (work in progress), November 2008.

[SIEVE] Guenther, P., Ed. and T. Showalter, Ed., "Sieve: An Email Filtering Language", RFC 5228, January 2008.

[StringPrep] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.

[TLS] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

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

[UTF-8] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[X509] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[X509-SRV] Santesson, S., "Internet X.509 Public Key Infrastructure Subject Alternative Name for Expression of Service Name", RFC 4985, August 2007.

9.2. Informative References

[DIGEST-MD5] Leach, P. and C. Newman, "Using Digest Authentication as a

SASL Mechanism", RFC 2831, May 2000.

[I-HAVE] Freed, N., "Sieve Email Filtering: Ihave Extension", draft-freed-sieve-ihave-03.txt (work in progress), October 2008.

[IMAP] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

[LDAP] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map", RFC 4510, June 2006.

[PLAIN] Zeilenga, K., "The PLAIN Simple Authentication and Security Layer (SASL) Mechanism", RFC 4616, August 2006.

Authors' Addresses

Alexey Melnikov (editor)
Isode Limited
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

Email: Alexey.Melnikov@isode.com

Tim Martin
BeThereBeSquare Inc.
672 Haight st.
San Francisco, CA 94117
US

Phone: +1 510 260-4175
Email: timmartin@alumni.cmu.edu