

INTERNET-DRAFT
Intended Category: Standards Track
Obsoletes: [4422](#) (if approved)
Category: Standards Track
Expires in six months

A. Melnikov, Ed.
Isode Limited
K. Zeilenga, Ed.
Isode Limited
14 April 2009

Simple Authentication and Security Layer (SASL)
<[draft-ietf-sasl-4422bis-01.txt](#)>

Status of This Memo

This document is intended to be, after appropriate review and revision, submitted to the RFC Editor as a Standards Track document. Distribution of this memo is unlimited. Technical discussion of this document will take place on the IETF SASL WG mailing list <ietf-sasl@imc.org>. Please send editorial comments directly to the editor.

This Internet-Draft is submitted to IETF in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/lid-abstracts.html>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

INTERNET-DRAFT

SASL

14 April 2008

Abstract

The Simple Authentication and Security Layer (SASL) is a framework for providing authentication and data security services in connection-oriented protocols via replaceable mechanisms. It provides a structured interface between protocols and mechanisms. The resulting framework allows new protocols to reuse existing mechanisms and allows old protocols to make use of new mechanisms. The framework also provides a protocol for securing subsequent protocol exchanges within a data security layer.

This document describes how a SASL mechanism is structured, describes how protocols include support for SASL, and defines the protocol for carrying a data security layer over a connection. In addition, this document defines one SASL mechanism, the EXTERNAL mechanism.

This document obsoletes [RFC 4422](#) [[when approved]].

INTERNET-DRAFT

SASL

14 April 2008

Table of Contents

[[Page numbers to be added by RFC-Editor]]

1. Introduction	
1.1. Document Audiences	
1.2. Relationship to Other Documents	
1.3. Conventions	
2. Identity Concepts	
3. The Authentication Exchange	
3.1. Mechanism Naming	
3.2. Mechanism Negotiation	
3.3. Request Authentication Exchange	
3.4. Challenges and Responses	
3.4.1. Authorization Identity String	
3.5. Aborting Authentication Exchanges	
3.6. Authentication Outcome	
3.7. Security Layers	
3.8. Multiple Authentications	
4. Protocol Requirements	
5. Mechanism Requirements	
6. Security Considerations	
6.1. Active Attacks	
6.1.1. Hijack Attacks	
6.1.2. Downgrade Attacks	
6.1.3. Replay Attacks	
6.1.4. Truncation Attacks	
6.1.5. Other Active Attacks	
6.2. Passive Attacks	
6.3. Re-keying	
6.4. Other Considerations	
7. IANA Considerations	
7.1. SASL Mechanism Registry	
7.2. Registration Changes	
8. References	
8.1. Normative References	

8.2. Informative References	
9. Acknowledgements	
Appendix A . The SASL EXTERNAL Mechanism	
A.1. EXTERNAL Technical Specification	
A.2. SASL EXTERNAL Examples	
A.3. Security Considerations	
Appendix B . Changes since RFC 4422	

INTERNET-DRAFT

SASL

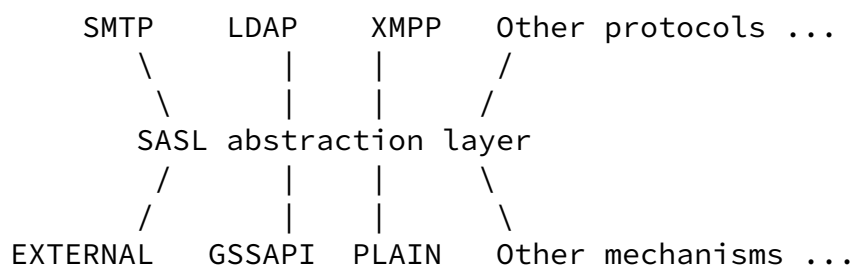
14 April 2008

[1](#). Introduction

The Simple Authentication and Security Layer (SASL) is a framework for providing authentication and data security services in connection-oriented protocols via replaceable mechanisms. SASL provides a structured interface between protocols and mechanisms. SASL also provides a protocol for securing subsequent protocol exchanges within a data security layer. The data security layer can provide data integrity, data confidentiality, and other services.

SASL's design is intended to allow new protocols to reuse existing mechanisms without requiring redesign of the mechanisms and allows existing protocols to make use of new mechanisms without redesign of protocols.

SASL is conceptually a framework that provides an abstraction layer between protocols and mechanisms as illustrated in the following diagram.



It is through the interfaces of this abstraction layer that the framework allows any protocol to utilize any mechanism. While this

layer does generally hide the particulars of protocols from mechanisms and the particulars of mechanisms from protocols, this layer does not generally hide the particulars of mechanisms from protocol implementations. For example, different mechanisms require different information to operate, some of them use password-based authentication, some of them require realm information, others make use of Kerberos tickets, certificates, etc. Also, in order to perform authorization, server implementations generally have to implement identity mapping between authentication identities, whose form is mechanism specific, and authorization identities, whose form is application protocol specific. [Section 2](#) discusses identity concepts.

It is possible to design and implement this framework in ways that do abstract away particulars of similar mechanisms. Such a framework implementation, as well as mechanisms implementations, could be designed not only to be shared by multiple implementations of a particular protocol but to be shared by implementations of multiple protocols.

The framework incorporates interfaces with both protocols and mechanisms in which authentication exchanges are carried out. [Section 3](#) discusses SASL authentication exchanges.

To use SASL, each protocol (amongst other items) provides a method for identifying which mechanism is to be used, a method for exchange of mechanism-specific server-challenges and client-responses, and a method for communicating the outcome of the authentication exchange. [Section 4](#) discusses SASL protocol requirements.

Each SASL mechanism defines (amongst other items) a series of server-challenges and client-responses that provide authentication services and negotiate data security services. [Section 5](#) discusses SASL mechanism requirements.

[Section 6](#) discusses security considerations. [Section 7](#) discusses IANA considerations. [Appendix A](#) defines the SASL EXTERNAL mechanism.

[1.1](#). Document Audiences

This document is written to serve several different audiences:

- protocol designers using this specification to support authentication in their protocol,
- mechanism designers that define new SASL mechanisms, and
- implementors of clients or servers for those protocols that support SASL.

While the document organization is intended to allow readers to focus on details relevant to their engineering, readers are encouraged to read and understand all aspects of this document.

[1.2.](#) Relationship to Other Documents

This document obsoletes [RFC 4422](#). It replaces [RFC 4422](#) in its entirety. [Appendix B](#) provides a summary of changes since [RFC 4422](#).

[1.3.](#) Conventions

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 [BCP 14](#) [[RFC2119](#)].

Character names in this document use the notation for code points and names from the Unicode Standard [[Unicode](#)]. For example, the letter "a" may be represented as either <U+0061> or <LATIN SMALL LETTER A>.

Note: a glossary of terms used in Unicode can be found in [[Glossary](#)]. Information on the Unicode character encoding model can be found in [[CharModel](#)].

In examples, "C:" and "S:" indicate lines of data to be sent by the client and server, respectively. Lines have been wrapped for improved readability.

[2.](#) Identity Concepts

In practice, authentication and authorization may involve multiple identities, possibly in different forms (simple username, Kerberos principal, X.500 Distinguished Name, etc.), possibly with different representations (e.g., ABNF-described UTF-8 encoded Unicode character string, BER-encoded Distinguished Name). While technical specifications often prescribe both the identity form and representation used on the network, different identity forms and/or representations may be (and often are) used within implementations. How identities of different forms relate to each other is, generally, a local matter. In addition, the forms and representations used within an implementation are a local matter.

However, conceptually, the SASL framework involves two identities:

- 1) an identity associated with the authentication credentials (termed the authentication identity), and
- 2) an identity to act as (termed the authorization identity).

SASL mechanism specifications describe the credential form(s) (e.g., X.509 certificates, Kerberos tickets, simple username/password) used to authenticate the client, including (where appropriate) the syntax and semantics of authentication identities carried in the credentials. SASL protocol specifications describe the identity form(s) used in authorization and, in particular, prescribe the syntax and semantics of the authorization identity character string to be transferred by mechanisms.

The client provides its credentials (which include or imply an authentication identity) and, optionally, a character string representing the requested authorization identity as part of the SASL exchange. When this character string is omitted or empty, the client is requesting to act as the identity associated with the credentials

(e.g., the user is requesting to act as the authentication identity).

The server is responsible for verifying the client's credentials and verifying that the identity it associates with the client's credentials (e.g., the authentication identity) is allowed to act as the authorization identity. A SASL exchange fails if either (or both) of these verifications fails. (The SASL exchange may fail for other reasons, such as service authorization failure.)

However, the precise form(s) of the authentication identities (used within the server in its verifications, or otherwise) and the precise form(s) of the authorization identities (used in making authorization decisions, or otherwise) are beyond the scope of SASL and this specification. In some circumstances, the precise identity forms used in some context outside of the SASL exchange may be dictated by other specifications. For instance, an identity assumption authorization (proxy authorization) policy specification may dictate how authentication and authorization identities are represented in policy statements.

[3.](#) The Authentication Exchange

Each authentication exchange consists of a message from the client to the server requesting authentication via a particular mechanism, followed by one or more pairs of challenges from the server and responses from the client, followed by a message from the server indicating the outcome of the authentication exchange. (Note: exchanges may also be aborted as discussed in [Section 3.5](#).)

The following illustration provides a high-level overview of an authentication exchange.

```
C: Request authentication exchange
S: Initial challenge
C: Initial response
<additional challenge/response messages>
S: Outcome of authentication exchange
```

If the outcome is successful and a security layer was negotiated, this layer is then installed (see [Section 3.7](#)). This also applies to the following illustrations.

Some mechanisms specify that the first data sent in the authentication exchange is from the client to the server. Protocols may provide an optional initial response field in the request message to carry this data. Where the mechanism specifies that the first data sent in the exchange is from the client to the server, the

protocol provides an optional initial response field, and the client

uses this field, the exchange is shortened by one round-trip:

C: Request authentication exchange + Initial response
<additional challenge/response messages>
S: Outcome of authentication exchange

Where the mechanism specifies that the first data sent in the exchange is from the client to the server and this field is unavailable or unused, the client request is followed by an empty challenge.

C: Request authentication exchange
S: Empty Challenge
C: Initial Response
<additional challenge/response messages>
S: Outcome of authentication exchange

Should a client include an initial response in its request where the mechanism does not allow the client to send data first, the authentication exchange fails.

Some mechanisms specify that the server is to send additional data to the client when indicating a successful outcome. Protocols may provide an optional additional data field in the outcome message to carry this data. Where the mechanism specifies that the server is to return additional data with the successful outcome, the protocol provides an optional additional data field in the outcome message, and the server uses this field, the exchange is shortened by one round-trip:

C: Request authentication exchange
S: Initial challenge
C: Initial response
<additional challenge/response messages>
S: Outcome of authentication exchange with
 additional data with success

Where the mechanism specifies that the server is to return additional data to the client with a successful outcome and this field is unavailable or unused, the additional data is sent as a challenge whose response is empty. After receiving this response, the server then indicates the successful outcome.

C: Request authentication exchange
S: Initial challenge
C: Initial response
<additional challenge/response messages>

S: Additional data challenge
C: Empty Response
S: Outcome of authentication exchange

Where mechanisms specify that the first data sent in the exchange is from the client to the server and additional data is sent to the client along with indicating a successful outcome, and the protocol provides fields supporting both, then the exchange takes two fewer round-trips:

C: Request authentication exchange + Initial response
<additional challenge/response messages>
S: Outcome of authentication exchange
with additional data with success

instead of:

C: Request authentication exchange
S: Empty Challenge
C: Initial Response
<additional challenge/response messages>
S: Additional data challenge
C: Empty Response
S: Outcome of authentication exchange

[3.1.](#) Mechanism Naming

SASL mechanisms are named by character strings, from 1 to 20 characters in length, consisting of ASCII [\[ASCII\]](#) uppercase letters, digits, hyphens, and/or underscores. In the following Augmented Backus-Naur Form (ABNF) [\[RFC5234\]](#) grammar, the <sasl-mech> production defines the syntax of a SASL mechanism name.

```
sasl-mech    = 1*20mech-char
mech-char    = UPPER-ALPHA / DIGIT / HYPHEN / UNDERSCORE
; mech-char is restricted to A-Z (uppercase only), 0-9, -, and _
; from ASCII character set.
```

```
UPPER-ALPHA  = %x41-5A ; A-Z (uppercase only)
DIGIT        = %x30-39 ; 0-9
HYPHEN       = %x2D ; hyphen (-)
UNDERSCORE   = %x5F ; underscore (_)
```

SASL mechanism names are registered as discussed in [Section 7.1](#).

[3.2.](#) Mechanism Negotiation

INTERNET-DRAFT

SASL

14 April 2008

Mechanism negotiation is protocol specific.

Commonly, a protocol will specify that the server advertises supported and available mechanisms to the client via some facility provided by the protocol, and the client will then select the "best" mechanism from this list that it supports and finds suitable.

Note that the mechanism negotiation is not protected by the subsequent authentication exchange and hence is subject to downgrade attacks if not protected by other means.

To detect downgrade attacks, a protocol can allow the client to discover available mechanisms subsequent to the authentication exchange and installation of data security layers with at least data integrity protection. This allows the client to detect changes to the list of mechanisms supported by the server.

[3.3.](#) Request Authentication Exchange

The authentication exchange is initiated by the client by requesting authentication via a mechanism it specifies. The client sends a message that contains the name of the mechanism to the server. The particulars of the message are protocol specific.

Note that the name of the mechanism is not protected by the mechanism, and hence is subject to alteration by an attacker if not integrity protected by other means.

Where the mechanism is defined to allow the client to send data first, and the protocol's request message includes an optional initial response field, the client may include the response to the initial challenge in the authentication request message.

[3.4.](#) Challenges and Responses

The authentication exchange involves one or more pairs of server-challenges and client-responses, the particulars of which are

mechanism specific. These challenges and responses are enclosed in protocol messages, the particulars of which are protocol specific.

Through these challenges and responses, the mechanism may:

- authenticate the client to the server,
- authenticate the server to the client,

- transfer an authorization identity string,
- negotiate a security layer, and
- provide other services.

The negotiation of the security layer may involve negotiation of the security services to be provided in the layer, how these services will be provided, and negotiation of a maximum cipher-text buffer size each side is able to receive in the layer (see [Section 3.6](#)).

After receiving an authentication request or any client response, the server may issue a challenge, abort the exchange, or indicate the outcome of an exchange. After receiving a challenge, a client mechanism may issue a response or abort the exchange.

[3.4.1](#). Authorization Identity String

The authorization identity string is a sequence of zero or more Unicode [[Unicode](#)] characters, excluding the NUL (U+0000) character, representing the identity to act as.

If the authorization identity string is absent, the client is requesting to act as the identity the server associates with the client's credentials. An empty string is equivalent to an absent authorization identity.

A non-empty authorization identity string indicates that the client wishes to act as the identity represented by the string. In this case, the form of identity represented by the string, as well as the precise syntax and semantics of the string, is protocol specific.

While the character encoding schema used to transfer the authorization identity string in the authentication exchange is mechanism specific, mechanisms are expected to be capable of carrying the entire Unicode repertoire (with the exception of the NUL character).

[3.5.](#) Aborting Authentication Exchanges

A client or server may desire to abort an authentication exchange if it is unwilling or unable to continue (or enter into).

A client may abort the authentication exchange by sending a message, the particulars of which are protocol specific, to the server, indicating that the exchange is aborted. The server may be required

by the protocol to return a message in response to the client's abort message.

Likewise, a server may abort the authentication exchange by sending a message, the particulars of which are protocol specific, to the client, indicating that the exchange is aborted.

[3.6.](#) Authentication Outcome

At the conclusion of the authentication exchange, the server sends a message, the particulars of which are protocol specific, to the client indicating the outcome of the exchange.

The outcome is not successful if

- the authentication exchange failed for any reason,
- the client's credentials could not be verified,
- the server cannot associate an identity with the client's credentials,
- the client-provided authorization identity string is malformed,

- the identity associated with the client's credentials is not authorized to act as the requested authorization identity,
- the negotiated security layer (or lack thereof) is not suitable, or
- the server is not willing to provide service to the client for any reason.

The protocol may include an optional additional data field in this outcome message. This field can only include additional data when the outcome is successful.

If the outcome is successful and a security layer was negotiated, this layer is then installed. If the outcome is unsuccessful, or a security layer was not negotiated, any existing security is left in place.

The outcome message provided by the server can provide a way for the client to distinguish between errors that are best dealt with by re-prompting the user for her credentials, errors that are best dealt with by telling the user to try again later, and errors where the user must contact a system administrator for resolution (see the SYS

and AUTH POP Response Codes [[RFC3206](#)] specification for an example). This distinction is particularly useful during scheduled server maintenance periods as it reduces support costs. It is also important that the server can be configured such that the outcome message will not distinguish between a valid user with invalid credentials and an invalid user.

[3.7.](#) Security Layers

SASL mechanisms may offer a wide range of services in security layers. Typical services include data integrity and data confidentiality. SASL mechanisms that do not provide a security layer are treated as negotiating no security layer.

If use of a security layer is negotiated in the authentication protocol exchange, the layer is installed by the server after indicating the outcome of the authentication exchange and installed

by the client upon receipt of the outcome indication. In both cases, the layer is installed before transfer of further protocol data. The precise position upon which the layer takes effect in the protocol data stream is protocol specific.

Once the security layer is in effect in the protocol data stream, it remains in effect until either a subsequently negotiated security layer is installed or the underlying transport connection is closed.

When in effect, the security layer processes protocol data into buffers of protected data. If at any time the security layer is unable or unwilling to continue producing buffers protecting protocol data, the underlying transport connection **MUST** be closed. If the security layer is not able to decode a received buffer, the underlying connection **MUST** be closed. In both cases, the underlying transport connection **SHOULD** be closed gracefully.

Each buffer of protected data is transferred over the underlying transport connection as a sequence of octets prepended with a four-octet field in network byte order that represents the length of the buffer. The length of the protected data buffer **MUST** be no larger than the maximum size that the other side expects. Upon the receipt of a length field whose value is greater than the maximum size, the receiver **SHOULD** close the connection, as this might be a sign of an attack.

The maximum size that each side expects is fixed by the mechanism, either through negotiation or by its specification.

[3.8.](#) Multiple Authentications

Unless explicitly permitted in the protocol (as stated in the protocol's technical specification), only one successful SASL authentication exchange may occur in a protocol session. In this case, once an authentication exchange has successfully completed, further attempts to initiate an authentication exchange fail.

Where multiple successful SASL authentication exchanges are permitted in the protocol, then in no case may multiple SASL security layers be simultaneously in effect. If a security layer is in effect and a subsequent SASL negotiation selects a second security layer, then the second security layer replaces the first. If a security layer is in

effect and a subsequent SASL negotiation selects no security layer, the original security layer remains in effect.

Where multiple successful SASL negotiations are permitted in the protocol, the effect of a failed SASL authentication exchange upon the previously established authentication and authorization state is protocol specific. The protocol's technical specification should be consulted to determine whether the previous authentication and authorization state remains in force, or changed to an anonymous state, or otherwise was affected. Regardless of the protocol-specific effect upon previously established authentication and authorization state, the previously negotiated security layer remains in effect.

4. Protocol Requirements

In order for a protocol to offer SASL services, its specification MUST supply the following information:

- 1) A service name, to be selected from registry of "service" elements for the Generic Security Service Application Program Interface (GSSAPI) host-based service name form, as described in [Section 4.1 of \[RFC2743\]](#). Note that this registry is shared by all GSSAPI and SASL mechanisms.
- 2) Detail any mechanism negotiation facility that the protocol provides (see [Section 3.2](#)).

A protocol SHOULD specify a facility through which the client may discover, both before initiation of the SASL exchange and after installing security layers negotiated by the exchange, the names of the SASL mechanisms that the server makes available to the client. The latter is important to allow the client to detect downgrade attacks. This facility is typically provided through the protocol's extensions or capabilities discovery facility.

- 3) Definition of the messages necessary for authentication exchange,

including the following:

- a) A message to initiate the authentication exchange (see [Section 3.3](#)).

This message MUST contain a field for carrying the name of the mechanism selected by the client.

This message SHOULD contain an optional field for carrying an initial response. If the message is defined with this field, the specification MUST describe how messages with an empty initial response are distinguished from messages with no initial response. This field MUST be capable of carrying arbitrary sequences of octets (including zero-length sequences and sequences containing zero-valued octets).

- b) Messages to transfer server challenges and client responses (see [Section 3.4](#)).

Each of these messages MUST be capable of carrying arbitrary sequences of octets (including zero-length sequences and sequences containing zero-valued octets).

- c) A message to indicate the outcome of the authentication exchange (see [Section 3.6](#)).

This message SHOULD contain an optional field for carrying additional data with a successful outcome. If the message is defined with this field, the specification MUST describe how messages with an empty additional data are distinguished from messages with no additional data. This field MUST be capable of carrying arbitrary sequences of octets (including zero-length sequences and sequences containing zero-valued octets).

- 4) Prescribe the syntax and semantics of non-empty authorization identity strings (see [Section 3.4.1](#)).

In order to avoid interoperability problems due to differing normalizations, the protocol specification MUST detail precisely how and where (client or server) non-empty authorization identity strings are prepared, including all normalizations, for comparison and other applicable functions to ensure proper function.

Specifications are encouraged to prescribe use of existing authorization identity forms as well as existing string representations, such as simple user names [[RFC4013](#)].

Where the specification does not precisely prescribe how identities in SASL relate to identities used elsewhere in the protocol, for instance, in access control policy statements, it may be appropriate for the protocol to provide a facility by which the client can discover information (such as the representation of the identity used in making access control decisions) about established identities for these uses.

- 5) Detail any facility the protocol provides that allows the client and/or server to abort authentication exchange (see [Section 3.5](#)).

Protocols that support multiple authentications typically allow a client to abort an ongoing authentication exchange by initiating a new authentication exchange. Protocols that do not support multiple authentications may require the client to close the connection and start over to abort an ongoing authentication exchange.

Protocols typically allow the server to abort ongoing authentication exchanges by returning a non-successful outcome message.

- 6) Identify precisely where newly negotiated security layers start to take effect, in both directions (see [Section 3.7](#)).

Typically, specifications require security layers to start taking effect on the first octet following the outcome message in data being sent by the server and on the first octet sent after receipt of the outcome message in data being sent by the client.

- 7) If the protocol supports other layered security services, such as Transport Layer Security (TLS) [[RFC5246](#)], the specification MUST prescribe the order in which security layers are applied to protocol data.

For instance, where a protocol supports both TLS and SASL security layers, the specification could prescribe any of the following:

- a) SASL security layer is always applied first to data being sent and, hence, applied last to received data,
- b) SASL security layer is always applied last to data being sent and, hence, applied first to received data,
- c) Layers are applied in the order in which they were installed,
- d) Layers are applied in the reverse order in which they were

installed, or

INTERNET-DRAFT

SASL

14 April 2008

e) Both TLS and SASL security layers cannot be installed.

- 8) Indicate whether the protocol supports multiple authentications (see [Section 3.8](#)). If so, the protocol MUST detail the effect a failed SASL authentication exchange will have upon a previously established authentication and authorization state.

Protocol specifications SHOULD avoid stating implementation requirements that would hinder replacement of applicable mechanisms. In general, protocol specifications SHOULD be mechanism neutral. There are a number of reasonable exceptions to this recommendation, including

- detailing how credentials (which are mechanism specific) are managed in the protocol,
- detailing how authentication identities (which are mechanism specific) and authorization identities (which are protocol specific) relate to each other, and
- detailing which mechanisms are applicable to the protocol.

[5.](#) Mechanism Requirements

SASL mechanism specifications MUST supply the following information:

- 1) The name of the mechanism (see [Section 3.1](#)). This name MUST be registered as discussed in [Section 7.1](#).
- 2) A definition of the server-challenges and client-responses of the authentication exchange, as well as the following:
 - a) An indication of whether the mechanism is client-first, variable, or server-first. If a SASL mechanism is defined as client-first and the client does not send an initial response in the authentication request, then the first server challenge MUST be empty (the EXTERNAL mechanism is an example of this case). If a SASL mechanism is defined as variable, then the specification needs to state how the server behaves when the

initial client response in the authentication request is omitted (the DIGEST-MD5 mechanism [DIGEST-MD5] is an example of this case). If a SASL mechanism is defined as server-first, then the client MUST NOT send an initial client response in the authentication request (the CRAM-MD5 mechanism [CRAM-MD5] is an example of this case).

- b) An indication of whether the server is expected to provide

additional data when indicating a successful outcome. If so, if the server sends the additional data as a challenge, the specification MUST indicate that the response to this challenge is an empty response.

SASL mechanisms SHOULD be designed to minimize the number of challenges and responses necessary to complete the exchange.

- 3) An indication of whether the mechanism is capable of transferring authorization identity strings (see [Section 3.4.1](#)). While some legacy mechanisms are incapable of transmitting an authorization identity (which means that for these mechanisms, the authorization identity is always the empty string), newly defined mechanisms SHOULD be capable of transferring authorization identity strings. The mechanism SHOULD NOT be capable of transferring both no authorization identity string and an empty authorization identity.

Mechanisms that are capable of transferring an authorization identity string MUST be capable of transferring arbitrary non-empty sequences of Unicode characters, excluding those that contain the NUL (U+0000) character. Mechanisms SHOULD use the UTF-8 [[RFC3629](#)] transformation format. The specification MUST detail how any Unicode code points special to the mechanism that might appear in the authorization identity string are escaped to avoid ambiguity during decoding of the authorization identity string. Typically, mechanisms that have special characters require these special characters to be escaped or encoded in the character string (after encoding it in a particular Unicode transformation format) using a data encoding scheme such as Base64 [[RFC4648](#)].

- 4) The specification MUST detail whether the mechanism offers a security layer. If the mechanism does, the specification MUST

detail the security and other services offered in the layer as well as how these services are to be implemented.

- 5) If the underlying cryptographic technology used by a mechanism supports data integrity, then the mechanism specification **MUST** integrity protect the transmission of an authorization identity and the negotiation of the security layer.

SASL mechanisms **SHOULD** be protocol neutral.

SASL mechanisms **SHOULD** reuse existing credential and identity forms, as well as associated syntaxes and semantics.

SASL mechanisms **SHOULD** use the UTF-8 transformation format [[RFC3629](#)] for encoding Unicode [[Unicode](#)] code points for transfer.

In order to avoid interoperability problems due to differing normalizations, when a mechanism calls for character data (other than the authorization identity string) to be used as input to a cryptographic and/or comparison function, the specification **MUST** detail precisely how and where (client or server) the character data is to be prepared, including all normalizations, for input into the function to ensure proper operation.

For simple user names and/or passwords in authentication credentials, SASLprep [[RFC4013](#)] (a profile of the StringPrep [[RFC3454](#)] preparation algorithm), **SHOULD** be specified as the preparation algorithm.

The mechanism **SHOULD NOT** use the authorization identity string in generation of any long-term cryptographic keys or hashes as there is no requirement that the authorization identity string be canonical. Long-term, here, means a term longer than the duration of the authentication exchange in which they were generated. That is, as different clients (of the same or different protocol) may provide different authorization identity strings that are semantically equivalent, use of authorization identity strings in generation of cryptographic keys and hashes will likely lead to interoperability and other problems.

[6.](#) Security Considerations

Security issues are discussed throughout this memo.

Many existing SASL mechanisms do not provide adequate protection against passive attacks, let alone active attacks, in the authentication exchange. Many existing SASL mechanisms do not offer security layers. It is hoped that future SASL mechanisms will provide strong protection against passive and active attacks in the authentication exchange, as well as security layers with strong basic data security features (e.g., data integrity and data confidentiality) services. It is also hoped that future mechanisms will provide more advanced data security services like re-keying (see [Section 6.3](#)).

Regardless, the SASL framework is susceptible to downgrade attacks. [Section 6.1.2](#) offers a variety of approaches for preventing or detecting these attacks. In some cases, it is appropriate to use data integrity protective services external to SASL (e.g., TLS) to protect against downgrade attacks in SASL. Use of external protective security services is also important when the mechanisms available do not themselves offer adequate integrity and/or confidentiality protection of the authentication exchange and/or protocol data.

[6.1.](#) Active Attacks

[6.1.1.](#) Hijack Attacks

When the client selects a SASL security layer with at least integrity protection, this protection serves as a counter-measure against an active attacker hijacking the connection and modifying protocol data sent after establishment of the security layer. Implementations SHOULD close the connection when the security services in a SASL security layer report protocol data report lack of data integrity.

[6.1.2.](#) Downgrade Attacks

It is important that any security-sensitive protocol negotiations be performed after installation of a security layer with data integrity protection. Protocols should be designed such that negotiations performed prior to this installation should be revalidated after installation is complete. Negotiation of the SASL mechanism is

security sensitive.

When a client negotiates the authentication mechanism with the server and/or other security features, it is possible for an active attacker to cause a party to use the least secure security services available. For instance, an attacker can modify the server-advertised mechanism list or can modify the client-advertised security feature list within a mechanism response. To protect against this sort of attack, implementations SHOULD NOT advertise mechanisms and/or features that cannot meet their minimum security requirements, SHOULD NOT enter into or continue authentication exchanges that cannot meet their minimum security requirements, and SHOULD verify that completed authentication exchanges result in security services that meet their minimum security requirements. Note that each endpoint needs to independently verify that its security requirements are met.

In order to detect downgrade attacks to the least (or less) secure mechanism supported, the client can discover the SASL mechanisms that the server makes available both before the SASL authentication exchange and after the negotiated SASL security layer (with at least data integrity protection) has been installed through the protocol's mechanism discovery facility. If the client finds that the integrity-protected list (the list obtained after the security layer was installed) contains a stronger mechanism than those in the previously obtained list, the client should assume that the previously obtained list was modified by an attacker and SHOULD close the underlying transport connection.

The client's initiation of the SASL exchange, including the selection

of a SASL mechanism, is done in the clear and may be modified by an active attacker. It is important for any new SASL mechanisms to be designed such that an active attacker cannot obtain an authentication with weaker security properties by modifying the SASL mechanism name and/or the challenges and responses.

Multi-level negotiation of security features is prone to downgrade attack. Protocol designers should avoid offering higher-level negotiation of security features in protocols (e.g., above SASL mechanism negotiation) and mechanism designers should avoid lower-level negotiation of security features in mechanisms (e.g., below SASL mechanism negotiation).

[6.1.3.](#) Replay Attacks

Some mechanisms may be subject to replay attacks unless protected by external data security services (e.g., TLS).

[6.1.4.](#) Truncation Attacks

Most existing SASL security layers do not themselves offer protection against truncation attack. In a truncation attack, the active attacker causes the protocol session to be closed, causing a truncation of the possibly integrity-protected data stream that leads to behavior of one or both the protocol peers that inappropriately benefits the attacker. Truncation attacks are fairly easy to defend against in connection-oriented application-level protocols. A protocol can defend against these attacks by ensuring that each information exchange has a clear final result and that each protocol session has a graceful closure mechanism, and that these are integrity protected.

[6.1.5.](#) Other Active Attacks

When use of a security layer is negotiated by the authentication protocol exchange, the receiver SHOULD handle gracefully any protected data buffer larger than the defined/negotiated maximal size. In particular, it MUST NOT blindly allocate the amount of memory specified in the buffer size field, as this might cause the "out of memory" condition. If the receiver detects a large block, it SHOULD close the connection.

[6.2.](#) Passive Attacks

Many mechanisms are subject to various passive attacks, including simple eavesdropping of unprotected credential information as well as online and offline dictionary attacks of protected credential information.

[6.3.](#) Re-keying

The secure or administratively permitted lifetimes of SASL mechanisms' security layers are finite. Cryptographic keys weaken as they are used and as time passes; the more time and/or cipher-text that a cryptanalyst has after the first use of the a key, the easier it is for the cryptanalyst to mount attacks on the key.

Administrative limits on a security layer's lifetime may take the form of time limits expressed in X.509 certificates, in Kerberos V tickets, or in directories, and are often desired. In practice, one likely effect of administrative lifetime limits is that applications may find that security layers stop working in the middle of application protocol operation, such as, perhaps, during large data transfers. As the result of this, the connection will be closed (see [Section 3.7](#)), which will result in an unpleasant user experience.

Re-keying (key renegotiation process) is a way of addressing the weakening of cryptographic keys. The SASL framework does not itself provide for re-keying; SASL mechanisms may. Designers of future SASL mechanisms should consider providing re-keying services.

Implementations that wish to re-key SASL security layers where the mechanism does not provide for re-keying SHOULD reauthenticate the same IDs and replace the expired or soon-to-expire security layers. This approach requires support for reauthentication in the application protocols (see [Section 3.8](#)).

[6.4.](#) Other Considerations

Protocol designers and implementors should understand the security considerations of mechanisms so they may select mechanisms that are applicable to their needs.

Distributed server implementations need to be careful in how they trust other parties. In particular, authentication secrets should only be disclosed to other parties that are trusted to manage and use those secrets in a manner acceptable to the disclosing party. Applications using SASL assume that SASL security layers providing data confidentiality are secure even when an attacker chooses the text to be protected by the security layer. Similarly, applications

assume that the SASL security layer is secure even if the attacker can manipulate the cipher-text output of the security layer. New SASL mechanisms are expected to meet these assumptions.

Unicode security considerations [UTR36] apply to authorization identity strings, as well as UTF-8 [RFC3629] security considerations where UTF-8 is used. SASLprep [RFC4013] and StringPrep [RFC3454] security considerations also apply where used.

7. IANA Considerations

7.1. SASL Mechanism Registry

The SASL mechanism registry is maintained by IANA. The registry is currently available at <<http://www.iana.org/assignments/sasl-mechanisms>>.

The purpose of this registry is not only to ensure uniqueness of values used to name SASL mechanisms, but also to provide a definitive reference to technical specifications detailing each SASL mechanism available for use on the Internet.

There is no naming convention for SASL mechanisms; any name that conforms to the syntax of a SASL mechanism name can be registered.

The procedure detailed in [Section 7.1.1](#) is to be used for registration of a value naming a specific individual mechanism.

The procedure detailed in [Section 7.1.2](#) is to be used for registration of a value naming a family of related mechanisms.

Comments may be included in the registry as discussed in [Section 7.1.3](#) and may be changed as discussed in [Section 7.1.4](#).

The SASL mechanism registry has been updated to reflect that this document provides the definitive technical specification for SASL and that this section provides the registration procedures for this registry.

7.1.1. Mechanism Name Registration Procedure

IANA will register new SASL mechanism names on a First Come First Served basis, as defined in [BCP 26](#) [RFC5226]. IANA has the right to reject obviously bogus registration requests, but will perform no review of claims made in the registration form.

INTERNET-DRAFT

SASL

14 April 2008

Registration of a SASL mechanism is requested by filling in the following template:

Subject: Registration of SASL mechanism X

SASL mechanism name (or prefix for the family):

Security considerations:

Published specification (recommended):

Person & email address to contact for further information:

Intended usage: (One of COMMON, LIMITED USE, or OBSOLETE)

Owner/Change controller:

Note: (Any other information that the author deems relevant may be added here.)

and sending it via electronic mail to IANA at <iana@iana.org>.

While this registration procedure does not require expert review, authors of SASL mechanisms are encouraged to seek community review and comment whenever that is feasible. Authors may seek community review by posting a specification of their proposed mechanism as an Internet-Draft. SASL mechanisms intended for widespread use should be standardized through the normal IETF process, when appropriate.

[7.1.2.](#) Family Name Registration Procedure

As noted above, there is no general naming convention for SASL mechanisms. However, specifications may reserve a portion of the SASL mechanism namespace for a set of related SASL mechanisms, a "family" of SASL mechanisms. Each family of SASL mechanisms is identified by a unique prefix, such as X-. Registration of new SASL mechanism family names requires expert review as defined in [BCP 26 \[RFC5226\]](#).

Registration of a SASL family name is requested by filling in the following template:

Subject: Registration of SASL mechanism family X

SASL family name (or prefix for the family):

Security considerations:

INTERNET-DRAFT

SASL

14 April 2008

Published specification (recommended):

Person & email address to contact for further information:

Intended usage: (One of COMMON, LIMITED USE, or OBSOLETE)

Owner/Change controller:

Note: (Any other information that the author deems relevant may be added here.)

and sending it via electronic mail to the IETF SASL mailing list at <ietf-sasl@imc.org> and carbon copying IANA at <iana@iana.org>. After allowing two weeks for community input on the IETF SASL mailing list, the expert will determine the appropriateness of the registration request and either approve or disapprove the request with notice to the requestor, the mailing list, and IANA.

The review should focus on the appropriateness of the requested family name for the proposed use and the appropriateness of the proposed naming and registration plan for existing and future mechanism names in the family. The scope of this request review may entail consideration of relevant aspects of any provided technical specification, such as their IANA Considerations section. However, this review is narrowly focused on the appropriateness of the requested registration and not on the overall soundness of any provided technical specification.

Authors are encouraged to pursue community review by posting the technical specification as an Internet-Draft and soliciting comment by posting to appropriate IETF mailing lists.

[7.1.3.](#) Comments on SASL Mechanism Registrations

Comments on a registered SASL mechanism/family should first be sent to the "owner" of the mechanism/family and/or to the <ietf-sasl@imc.org> mailing list.

Submitters of comments may, after a reasonable attempt to contact the owner, request IANA to attach their comment to the SASL mechanism registration itself by sending mail to <iana@iana.org>. At IANA's sole discretion, IANA may attach the comment to the SASL mechanism's registration.

[7.1.4.](#) Change Control

Once a SASL mechanism registration has been published by IANA, the author may request a change to its definition. The change request follows the same procedure as the registration request.

The owner of a SASL mechanism may pass responsibility for the SASL mechanism to another person or agency by informing IANA; this can be done without discussion or review.

The IESG may reassign responsibility for a SASL mechanism. The most common case of this will be to enable changes to be made to mechanisms where the author of the registration has died, has moved out of contact, or is otherwise unable to make changes that are important to the community.

SASL mechanism registrations may not be deleted; mechanisms that are no longer believed appropriate for use can be declared OBSOLETE by a change to their "intended usage" field; such SASL mechanisms will be clearly marked in the lists published by IANA.

The IESG is considered to be the owner of all SASL mechanisms that are on the IETF standards track.

[7.2.](#) Registration Changes

The IANA has updated the SASL mechanisms registry as follows:

- 1) Changed the "Intended usage" of the KERBEROS_V4 and SKEY mechanism

registrations to OBSOLETE.

- 2) Changed the "Published specification" of the EXTERNAL mechanism to this document as indicated below:

Subject: Updated Registration of SASL mechanism EXTERNAL
Family of SASL mechanisms: NO
SASL mechanism name: EXTERNAL
Security considerations: See A.3 of [RFC 4422](#)
Published specification (optional, recommended): [RFC 4422](#)
Person & email address to contact for further information:
 Alexey Melnikov <Alexey.Melnikov@isode.com>
Intended usage: COMMON
Owner/Change controller: IESG <iesg@ietf.org>
Note: Updates existing entry for EXTERNAL

[8.](#) References

[8.1.](#) Normative References

Melnikov & Zeilenga [draft-ietf-sasl-4422bis-01](#) [Page 26]

INTERNET-DRAFT SASL 14 April 2008

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", [RFC 2743](#), January 2000.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, [RFC 3629](#), November 2003.
- [RFC4013] Zeilenga, K., "SASLprep: Stringprep Profile for User Names and Passwords", [RFC 4013](#), February 2005.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 5226](#), May 1998.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, [RFC 5234](#), January 2008.
- [ASCII] Coded Character Set--7-bit American Standard Code for

Information Interchange, ANSI X3.4-1986.

- [Unicode] The Unicode Consortium, "The Unicode Standard, Version 3.2.0" is defined by "The Unicode Standard, Version 3.0" (Reading, MA, Addison-Wesley, 2000. ISBN 0-201-61633-5), as amended by the "Unicode Standard Annex #27: Unicode 3.1" (<http://www.unicode.org/reports/tr27/>) and by the "Unicode Standard Annex #28: Unicode 3.2" (<http://www.unicode.org/reports/tr28/>).
- [CharModel] Whistler, K. and M. Davis, "Unicode Technical Report #17, Character Encoding Model", UTR17, <<http://www.unicode.org/unicode/reports/tr17/>>, August 2000.

8.2. Informative References

- [RFC2244] Newman, C. and J. G. Myers, "ACAP -- Application Configuration Access Protocol", [RFC 2244](#), November 1997.
- [RFC3206] Gellens, R., "The SYS and AUTH POP Response Codes", [RFC 3206](#), February 2002.
- [RFC3454] Hoffman, P. and M. Blanchet, "Preparation of

- Internationalized Strings ("stringprep")", [RFC 3454](#), December 2002.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", [RFC 4301](#), December 2005.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", [RFC 4648](#), October 2006.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), August 2008.
- [UTR36] Davis, M. and Michel Suignard, "Unicode Technical

Report #36, Unicode Security Considerations", UTR36,
<<http://www.unicode.org/unicode/reports/tr36/>>, July
2008.

[Glossary] The Unicode Consortium, "Unicode Glossary",
<<http://www.unicode.org/glossary/>>.

9. Acknowledgements

This document is a revision of [RFC 4422](#), a product of the IETF Simple Authentication and Security Layer (SASL) Working Group. [RFC 4422](#) was a revision of [RFC 2222](#) by John Myers.

This revision is also a product of the IETF SASL Working Group.

The following individuals contributed significantly to this revision:
TBD

Appendix A. The SASL EXTERNAL Mechanism

This appendix is normative.

The EXTERNAL mechanism allows a client to request the server to use credentials established by means external to the mechanism to authenticate the client. The external means may be, for instance, IP

Security [[RFC4301](#)] or TLS [[RFC5246](#)] services. In absence of some a priori agreement between the client and the server, the client cannot make any assumption as to what external means the server has used to obtain the client's credentials, nor make an assumption as to the form of credentials. For example, the client cannot assume that the server will use the credentials the client has established via TLS.

[A.1.](#) EXTERNAL Technical Specification

The name of this mechanism is "EXTERNAL".

The mechanism does not provide a security layer.

The mechanism is capable of transferring an authorization identity string. If empty, the client is requesting to act as the identity the server has associated with the client's credentials. If non-empty, the client is requesting to act as the identity represented by the string.

The client is expected to send data first in the authentication exchange. Where the client does not provide an initial response data in its request to initiate the authentication exchange, the server is to respond to the request with an empty initial challenge and then the client is to provide its initial response.

The client sends the initial response containing the UTF-8 [[RFC3629](#)] encoding of the requested authorization identity string. This response is non-empty when the client is requesting to act as the identity represented by the (non-empty) string. This response is empty when the client is requesting to act as the identity the server associated with its authentication credentials.

The syntax of the initial response is specified as a value of the <extern-initial-resp> production detailed below using the Augmented Backus-Naur Form (ABNF) [[RFC5234](#)] notation.

```
external-initial-resp = authz-id-string
authz-id-string       = *( UTF8-char-no-nul )
UTF8-char-no-nul      = UTF8-1-no-nul / UTF8-2 / UTF8-3 / UTF8-4
UTF8-1-no-nul         = %x01-7F
```

where the <UTF8-2>, <UTF8-3>, and <UTF8-4> productions are as defined in [[RFC3629](#)].

There are no additional challenges and responses.

Hence, the server is to return the outcome of the authentication exchange.

The exchange fails if

- the client has not established its credentials via external means,
- the client's credentials are inadequate,
- the client provided an empty authorization identity string and the server is unwilling or unable to associate an authorization identity with the client's credentials,
- the client provided a non-empty authorization identity string that is invalid per the syntax requirements of the applicable application protocol specification,
- the client provided a non-empty authorization identity string representing an identity that the client is not allowed to act as, or
- the server is unwilling or unable to provide service to the client for any other reason.

Otherwise the exchange is successful. When indicating a successful outcome, additional data is not provided.

[A.2.](#) SASL EXTERNAL Examples

This section provides examples of EXTERNAL authentication exchanges. The examples are intended to help the readers understand the above text. The examples are not definitive. The Application Configuration Access Protocol (ACAP) [[RFC2244](#)] is used in the examples.

The first example shows use of EXTERNAL with an empty authorization identity. In this example, the initial response is not sent in the client's request to initiate the authentication exchange.

```
S: * ACAP (SASL "GSSAPI")
C: a001 STARTTLS
S: a001 OK "Begin TLS negotiation now"
```

INTERNET-DRAFT

SASL

14 April 2008

```
<TLS negotiation, further commands are under TLS layer>
S: * ACAP (SASL "GSSAPI" "EXTERNAL")
C: a002 AUTHENTICATE "EXTERNAL"
S: + ""
C: + ""
S: a002 OK "Authenticated"
```

The second example shows use of EXTERNAL with an authorization identity of "fred@example.com". In this example, the initial response is sent with the client's request to initiate the authentication exchange. This saves a round-trip.

```
S: * ACAP (SASL "GSSAPI")
C: a001 STARTTLS
S: a001 OK "Begin TLS negotiation now"
<TLS negotiation, further commands are under TLS layer>
S: * ACAP (SASL "GSSAPI" "EXTERNAL")
C: a002 AUTHENTICATE "EXTERNAL" {16+}
C: fred@example.com
S: a002 NO "Cannot assume requested authorization identity"
```

[A.3.](#) Security Considerations

The EXTERNAL mechanism provides no security protection; it is vulnerable to spoofing by either client or server, active attack, and eavesdropping. It should only be used when adequate security services have been established.

[Appendix B.](#) Changes since [RFC 4422](#)

This appendix is non-normative.

The following is a summary of the changes were made:

- References updated and corrected.

Editors' Addresses

Alexey Melnikov
Isode Limited
5 Castle Business Village

36 Station Road
Hampton, Middlesex TW12 2BX
UK

EMail: Alexey.Melnikov@isode.com

INTERNET-DRAFT

SASL

14 April 2008

Kurt D. Zeilenga
Isode Limited

EMail: Kurt.Zeilenga@isode.com

Full Copyright

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (<http://trustee.ietf.org/license-info>). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

