# XEP-0034: SASL Integration


Robert Norris
mailto:rob@cataclysm.cx
xmpp:rob@cataclysm.cx

Jeremie Miller
mailto:jeremie@jabber.org
xmpp:jer@jabber.org

Peter Saint-Andre
mailto:stpeter@jabber.org
xmpp:stpeter@jabber.org



2003-11-05
Version 1.1

| Status | Type | Short Name |
| --- | --- | --- |
| Retracted | Standards Track | N/A |


NOTE WELL: this specification was retracted on 2003-11-05 since the topic is addressed definitively in XMPP Core. Please refer to XMPP Core for further information.

# Legal

## Copyright



## Permissions



## Warranty



## Liability



## Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF’s Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

# Contents

# 1 Introduction

The Simple Authentication and Security Layer (SASL) (see RFC 4422 [1]) provides a generalized method for adding authentication support to connection-based protocols. This document describes a generic XML namespace profile for SASL, that conforms to section 4 of RFC 4422, "Profiling requirements".
This profile may be used for both client-to-server and server-to-server connections. For client connections, the service name used is "jabber-client". For server connections, the service name used is "jabber-server". Both these names are registered in the IANA service registry.
The reader is expected to have read and understood the SASL specification before reading this document.

# 2 Protocol

## 2.1 Overview

In these examples, "client" refers to the remote entity that initiated the connection, either a Jabber client or a Jabber server. "Server" refers to the server that the remote entity is attempting to connect and authenticate to.
The steps involved for a SASL negotiation are as follows:

1. Client requests SASL authentication
2. Server responds with list of available SASL authentication mechanisms
3. Client selects mechanism
4. Server sends a challenge
5. Client responds to challenge
6. Server sends more challenges, client sends more responses

This series of challenge/response pairs continues until one of three things happens:

1. Client aborts the handshake.
2. Server reports failure.
3. Server reports success.

After authentication has completed, the client sends a packet to begin the session.
The namespace identifier for this protocol is`http://www.iana.org/assignments/sasl-mechanisms`.\\ The following examples show the dialogue between a client [C] and a server [S].

[1] RFC 4422: Simple Authentication and Security Layer (SASL) <http://tools.ietf.org/html/rfc4422>.

## 2.2 Stream initialization

The client begins by requesting SASL authentication as part of the normal Jabber stream negotiation. [2] The server responds by sending the available authentication mechanisms to the client along with the stream information:

Listing 1: Stream initialization

```
C: <stream:stream xmlns='jabber:client'
                  xmlns:stream='http://etherx.jabber.org/streams'
                  xmlns:sasl='http://www.iana.org/assignments/sasl-
                      mechanisms'
                  to='jabber.org'>
S: <stream:stream xmlns='jabber:client'
                  xmlns:stream='http://etherx.jabber.org/streams'
                  xmlns:sasl='http://www.iana.org/assignments/sasl-
                      mechanisms'
                  id='12345678'>
   <sasl:mechanisms>
     <sasl:mechanism>PLAIN</sasl:mechanism>
     <sasl:mechanism>DIGEST-MD5</sasl:mechanism>
     <sasl:mechanism>EXTERNAL</sasl:mechanism>
   <sasl:mechanisms>
```

## 2.3 Mechanism selection and handshake

Next, the client selects an authentication mechanism:

Listing 2: Plaintext mechanism selection

```
C: <sasl:auth mechanism='PLAIN'/>
```

The server responds with a mechanism-specific challenge, which the client must respond to. More than one challenge/response pair can take place; this is mechanism-specific. Challenges and responses are Base64[3] encoded.

Listing 3: Plaintext handshake

```
S: <sasl:challenge/>
C: <sasl:response>cm9iAHNlY3JldA==</sasl:response>
```

Listing 4: Digest handshake

```
S: <sasl:challenge>
```

[2] In the case of the remote entity being a server, the default namespace in the stream header will be "jabber:server".

[3] RFC 2045, section 6.8.

```
      cmVhbG09ImNhdGFjbHlzbS5jeCIsbm9uY2U9Ik9BNk1HOXRFUUdtMmhoIi
      xxb3A9ImF1dGgiLGNoYXJzZXQ9dXRmLTgsYWxnb3JpdGhtPW1kNS1zZXNz
    </sasl:challenge>
C: <sasl:response>
      dXNlcm5hbWU9InJvYiIscmVhbG09ImNhdGFjbHlzbS5jeCIsbm9uY2U9Ik
      9BNk1HOXRFUUdtMmhoIixjbm9uY2U9Ik9BNk1IWGg2VnFUclJrIixuYz0w
      MDAwMDAwMSxxb3A9YXV0aCxkaWdlc3QtdXJpPSJqYWJiZXIvY2F0YWNseX
      NtLmN4IixyZXNwb25zZT1kMzg4ZGFkOTBkNGJiZDc2MGExNTIzMjFmMjE0
      M2FmNyxjaGFyc2V0PXV0Zi04
    </sasl:response>
S: <sasl:challenge>
      cnNwYXV0aD1lYTQwZjYwMzM1YzQyN2I1NTI3Yjg0ZGJhYmNkZmZmZA==
    </sasl:challenge>
C: <sasl:response/>
```

For mechanisms that require the client to send data first (ie the first challenge from the server is empty), the client may optionally send its first response as part of the mechanism selection:

Listing 5: Plaintext mechanism selection; client sends data first

```
C: <sasl:auth mechanism='PLAIN'>cm9iAHNlY3JldA==</sasl:auth>
```

## 2.4 Success, failure and client abort

The handshake continues until authentication completes successfully, authentication fails, or the client aborts the handshake:

Listing 6: Authentication success

```
S: <sasl:success/>
```

Listing 7: Authentication failure

```
S: <sasl:failure/>
```

Listing 8: Client abort

```
C: <sasl:abort/>
```

Optionally, the server or client may send an informative message along with the success, failure or abort command:

Listing 9: Authentication failure; optional informative message

```
S: <sasl:failure>Plaintext authentication failed (Incorrect username
   or password)</sasl:failure>
```

Following a failure or client abort, the client may start a new handshake. Following a successful authentication, any further attempts by the client to begin a new authentication handshake will automatically result in the server sending a failure.

### 2.5 Session start

Note: that this section only applies to client-to-server connections.
Following successful authentication, the client must send a standard IQ set packet in the jabber:iq:auth namespace to start a session. The client must supply a username and resource for the session along with this packet.

Listing 10: Session start after successful authentication

```
C: <iq id="a1" type="get">
     <query xmlns="jabber:iq:auth">
       <username>rob</username>
     </query>
   </iq>
S: <iq id="a1" type="result">
     <query xmlns="jabber:iq:auth">
       <username>rob</username>
       <resource/>
     </query>
   </iq>
C: <iq id="a2" type="set">
     <query xmlns="jabber:iq:auth">
       <username>rob</username>
       <resource>laptop</resource>
     </query>
   </iq>
S: <iq id="a2" type="result">
     <query xmlns="jabber:iq:auth"/>
   </iq>
```

If the client attempts to start a session before authenticating, or the username given in the jabber:iq:auth packet does not match the username given in the authentication credentials (when the SASL mechanism supports it), the server will return a 401 (Unauthorized) error packet.

## 3 Support for existing authentication methods

Traditionally, Jabber servers have supported two authentication models - jabber:iq:auth for client-to-server authentication, and dialback for server-to-server authentication.

## 3.1 Legacy client-to-server authentication support

Until SASL authentication is in widespread use, clients and servers may support both SASL and the legacy jabber:iq:auth authentication system for client-to-server connections. Note that neither the client nor the server are required to support legacy authentication; it is simply a courtesy to users until the majority of clients and servers support SASL authentication.
If a client connects and does not request the use of SASL (that is, the SASL profile namespace identifier does not appear in the stream initializer response), then the server should disable SASL for this connection; that is, it should not add the SASL profile namespace identifier to the stream initialization response, nor should it offer any SASL mechanisms.
If a client connects to a server that does not support SASL (identified by the lack of the SASL profile namespace identifier in the stream initializer response, even though the client requested it), the client may choose to fall back to use legacy authentication.

## 3.2 Dialback support for server-to-server authentication

SASL authentication for server-to-server connections is not intended to replace dialback, as there are uses for both. Dialback is useful in an uncontrolled environment, such as the global Internet, where it is necessary to verify the identity of the remote server. SASL authentication has uses in a more controlled environment, where the administrator wishes to restrict access to a certain number of known remote servers.
To this end, the use of dialback is not deprecated. If a remote server connects and requests the use of dialback (by specifying the ”jabber:server:dialback” namespace, the the local server shall not offer SASL authentication. Similarly, if the remote server connects and requests the use of SASL authentication, then the local server shall not offer dialback. In the event that the remote server requests both, the local server should terminate the stream immediately and close the connection. If the remote server requests neither, then the local server may choose to support the pre-dialback server-to-server stream, but it is recommended that the local server terminate the stream and close the connection.