



# XMPP

## XEP-0509: Initial Authentication Pipelining

Dave Cridland

<mailto:dave@crdland.net>

<xmpp:dwd@dave.cridland.net>

2025-12-12

Version 0.1.0

Status	Type	Short Name
Experimental	Standards Track	NOT_YET_ASSIGNED

This specification defines a protocol for discovering if the SASL2 <authenticate> can be pipelined safely along with the stream open, and if so allows the client to perform this pipelining safely.

# Legal

## Copyright

This XMPP Extension Protocol is copyright © 1999 – 2024 by the [XMPP Standards Foundation](#) (XSF).

## Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the "Specification"), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

## Warranty

## NOTE WELL: This Specification is provided on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. ##

## Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

## 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

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Protocol</b>	<b>1</b>
2.1	Stream Feature . . . . .	1
2.2	Usage . . . . .	1
2.3	Config Versioning . . . . .	2
2.3.1	opaque . . . . .	2
<b>3</b>	<b>Examples</b>	<b>3</b>
<b>4</b>	<b>Security Considerations</b>	<b>6</b>
<b>5</b>	<b>XMPP Registrar Considerations</b>	<b>6</b>
5.1	Protocol Namespaces . . . . .	6
5.2	Stream Features . . . . .	6
<b>6</b>	<b>IANA Considerations</b>	<b>6</b>

## 1 Introduction

When clients (or initiating servers) connect, thanks to [Extensible SASL Profile \(XEP-0388\)](#)<sup>1</sup> there is now a single waited-for round-trip after the connection itself is established, before the exchange of stanzas can occur. This is because after sending the stream open, the initiator cannot know the features (and thus what SASL2 etc options are supported) until it has received them, and until it does receive them, therefore cannot know what the optimal features and suboptions are to negotiate.

Although specifications such as [Fast Authentication Streamlining Tokens \(XEP-0484\)](#)<sup>2</sup> suggest that pipelining is feasible, clients which do so risk "missing out" on improved security and other features.

This protocol, herein "IAP", therefore allows a server to provide a token which is tied to what features are available, which the client then uses to indicate the basis of its authentication attempt. If this does not match, the authentication attempt is rejected without prejudice, and can be retried with the currently available features.

Note that the protocol is explicitly possible for both C2S and S2S.

## 2 Protocol

### 2.1 Stream Feature

If a server supports IAP, this is indicated by a stream feature, consisting of an element of `config-version`, qualified by the namespace `urn:xmpp:iap:0`. This has two attributes, first, a optional scheme, which has the value (if present) of "opaque". Second, a mandatory value, for which the value is as defined below.

Listing 1: Stream features

```
<stream:features>
  <!--{}- ... -{}-->
  <config-version xmlns='urn:xmpp:iap:0' schema='opaque' value='
    V2VsbCwgb2YgY291cnNl' />
</stream:features>
```

### 2.2 Usage

When an initiator wishes to pipeline the `<authenticate>` with the stream open, it first caches the known config version, and sends this in a similar `<config-version>`, with the same optional scheme and value attributes as used by the responding entity, as a child of the `<authenticate>`. The responder then attempts to match this value against its own current value before proceeding - the precise meaning of match depends on the scheme, though a simple string

---

<sup>1</sup>XEP-0388: Extensible SASL Profile <<https://xmpp.org/extensions/xep-0388.html>>.

<sup>2</sup>XEP-0484: Fast Authentication Streamlining Tokens <<https://xmpp.org/extensions/xep-0484.html>>.

comparison MUST always be a match.

If a responder sees such an element and it does not match the current value, the responder rejects the <authenticate> with a <failure> "without prejudice", meaning that the attempted authentication is not counted as a "failed login", the account is not penalised or locked out, etc.

The failure SHOULD be of type <aborted>, and carry an application-specific error of <config-version-mismatch>, qualified by the 'urn:xmpp:iap:0' namespace.

Otherwise, the SASL2 authentication proceeds as per normal.

Listing 2: Example 4 of XEP-0388 with the config-version element included

```
<authenticate xmlns='urn:xmpp:sasl:2' mechanism='BLURDYBLOOP'>
  <initial-response>
    SSBzaG91bGQgbWFrZSB0aGlzIGEgY29tcGV0aXRpb24=
  </initial-response>
  <user-agent id='d4565fa7-4d72-4749-b3d3-740edbf87770'>
    <software>AwesomeXMPP</software>
    <device>Kiva's Phone</device>
  </user-agent>
  <bind xmlns='urn:xmpp:bind:example' />
  <config-version xmlns='urn:xmpp:iap:0' _schema='opaque' _value='
    Tm9uLW1hdGNoaW5n' />
</authenticate>
```

## 2.3 Config Versioning

Config versions have schemes, which denote how the server calculates the version. The only defined scheme in this specification (so far) is "opaque", and all token schemes are compatible with "opaque".

### 2.3.1 opaque

The "opaque" scheme is a token which the client MUST NOT (and cannot) derive any meaning from.

While the server is in principle free to include any structured data such that it can "match" a token even if the current configuration differs, typically it would be expected that additional SASL mechanisms, SASL2 subfeatures, etc would be cause to reject the token (as the client might have wished to negotiate these instead or as well). Therefore such "smart" matching is both out of scope for this specification and moreover - until more implementation experience is gained - NOT RECOMMENDED

An exact match (ie, identical tokens by "i;octet") with the currently advertised value MUST be treated as a match.

One example of a conformant mechanism for generating the value would be to generate a hash or HMAC of the rendered features, absent the <config-version>. Note that the initiating

entity MUST NOT assume this mechanism is in use, nor attempt to calculate any such hash itself.

### 3 Examples

Here is the flow from XEP-0484 (FAST), as a single example including the full stream and with annotations to indicate packets and round-trips.

Listing 3: Initial session

```
<!--{}- Client sends first --{}->
  <!--{}- C --{}-><stream:stream xmlns:stream='http://etherx.jabber.
    org/stream' xmlns='jabber:client' from='user@example.com' to='
    example.com' version='1.0'>
    <!--{}- Client now waits for stream features from server: --{}->
    <!--{}- S --{}-><stream:stream xmlns:stream='http://etherx.jabber.
      org/stream' xmlns='jabber:client' from='example.com' version='
      1.0'>
<stream:features>
  <authentication xmlns='urn:xmpp:sasl:2'>
    <mechanism>SCRAM-SHA-1</mechanism>
    <mechanism>SCRAM-SHA-1-PLUS</mechanism>
    <inline>
      <fast xmlns='urn:xmpp:fast:0' tls-0rtt='true'>
        <mechanism>HT-SHA-256-ENDP</mechanism>
        <mechanism>HT-SHA-256-EXPR</mechanism>
        <mechanism>HT-SHA-256-NONE</mechanism>
      </fast>
    </inline>
  </authentication>
  <config-version xmlns='urn:xmpp:iap:0' value='VGhpcyBpcyBvcGFxdWUh'/
  >
</stream:features>
<!--{}- Client can now authenticate, request a FAST token, etc. No need
  nor advantage to include the config version. --{}->
<!--{}- C --{}-><authenticate xmlns='urn:xmpp:sasl:2' mechanism='SCRAM-
  SHA-1-PLUS'>
  <initial-response>[base64 encoded SASL data]</initial-response>
  <bind xmlns='urn:xmpp:bind:0'>
    <tag>AwesomeXMPP</tag>
  </bind>
  <request-token xmlns='urn:xmpp:fast:0' mechanism='HT-SHA-256-ENDP' />
</authenticate>
<!--{}- ... Normal SASL2 exchange --{}->
<!--{}- S --{}-><success xmlns='urn:xmpp:sasl:2'>
  <authorization-identity>user@example.com/AwesomeXMPP.4232f4d4</
  authorization-identity>
```

```
<bound xmlns='urn:xmpp:bind:0'>
  <metadata xmlns='urn:xmpp:mam:2'>
    <start id='YWxwaGEg' timestamp='2008-08-22T21:09:04Z' />
    <end id='b21lZ2Eg' timestamp='2020-04-20T14:34:21Z' />
  </metadata>
</bound>
<token xmlns='urn:xmpp:fast:0'
  expiry='2020-03-12T14:36:15Z'
  token='WXZzciBwYmFmdmZnZiBqdmd1IGp2eXFhcmZm' />
</success>
```

Listing 4: Subsequent session

```
<!--{}- Client sends first, pipelining the authentication --{}->
  <!--{}- C --{}-><stream:stream xmlns:stream='http://etherx.jabber.
    org/stream' xmlns='jabber:client' from='user@example.com' to='
    example.com'>
    <authenticate xmlns='urn:xmpp:sasl:2' mechanism='HT-SHA-256-ENDP'>
    <initial-response>[base64 encoded SASL data]</initial-response>
    <bind xmlns='urn:xmpp:bind:0'>
      <tag>AwesomeXMPP</tag>
    </bind>
    <fast xmlns='urn:xmpp:fast:0' count='123' />
    <config-version xmlns='urn:xmpp:iap:0' scheme='opaque' value='
      VGhpcyBpcyBvcGFxdWUu' />
  </authenticate>
  <!--{}- Client now waits for stream features from server and
    result: --{}->
  <!--{}- S --{}-><stream:stream xmlns:stream='http://etherx.jabber.
    org/stream' xmlns='jabber:client' from='example.com' version='
    1.0'>
  <stream:features>
    <authentication xmlns='urn:xmpp:sasl:2'>
      <mechanism>SCRAM-SHA-1</mechanism>
      <mechanism>SCRAM-SHA-1-PLUS</mechanism>
      <inline>
        <fast xmlns='urn:xmpp:fast:0' tls-0rtt='true'>
          <mechanism>HT-SHA-256-ENDP</mechanism>
          <mechanism>HT-SHA-256-EXPR</mechanism>
          <mechanism>HT-SHA-256-NONE</mechanism>
        </fast>
      </inline>
    </authentication>
    <config-version xmlns='urn:xmpp:iap:0' value='VGhpcyBpcyBvcGFxdWUu' />
  </stream:features>
  <success xmlns='urn:xmpp:sasl:2'>
    <authorization-identity>user@example.com/AwesomeXMPP.4232f4d4</
      authorization-identity>
    <bound xmlns='urn:xmpp:bind:0'>
```

```

    <metadata xmlns='urn:xmpp:mam:2'>
      <start id='YWxwaGEg' timestamp='2008-08-22T21:09:04Z' />
      <end id='b21lZ2Eg' timestamp='2020-04-20T14:34:21Z' />
    </metadata>
  </bound>
  <token xmlns='urn:xmpp:fast:0'
    expiry='2020-03-31T14:36:15Z'
    token='R3VyIHpiZmcgbnl2aXIgdMYgZ3VyIGp2eXFyZmcu' />
</success>

```

Listing 5: Failed subsequent session

```

<!--{}- Client sends first, pipelining the authentication -{}-->
  <!--{}- C -{}--><stream:stream xmlns:stream='http://etherx.jabber.
    org/stream' xmlns='jabber:client' from='user@example.com' to='
    example.com'>
    <authenticate xmlns='urn:xmpp:sasl:2' mechanism='HT-SHA-256-ENDP'>
    <initial-response>[base64 encoded SASL data]</initial-response>
    <bind xmlns='urn:xmpp:bind:0'>
      <tag>AwesomeXMPP</tag>
    </bind>
    <fast xmlns='urn:xmpp:fast:0' count='123' />
    <config-version xmlns='urn:xmpp:iap:0' scheme='opaque' value='
      VGhpcyBpcyBvcGFxdWUu' />
  </authenticate>
  <!--{}- Client now waits for stream features from server and
    result: -{}-->
  <!--{}- S -{}--><stream:stream xmlns:stream='http://etherx.jabber.
    org/stream' xmlns='jabber:client' from='example.com' version='
    1.0'>
  <stream:features>
    <authentication xmlns='urn:xmpp:sasl:2'>
      <mechanism>SCRAM-SHA-1</mechanism>
      <mechanism>SCRAM-SHA-1-PLUS</mechanism>
      <inline>
        <fast xmlns='urn:xmpp:fast:0' tls-0rtt='true'>
          <mechanism>HT-SHA-256-ENDP</mechanism>
          <mechanism>HT-SHA-256-EXPR</mechanism>
          <mechanism>HT-SHA-256-NONE</mechanism>
        </fast>
      </inline>
    </authentication>
    <config-version xmlns='urn:xmpp:iap:0' value='SSBjaGFuZ2VkJHRoaXMh' />
  </stream:features>
  <failure xmlns='urn:xmpp:sasl:2'>
    <aborted xmlns='urn:ietf:params:xml:ns:xmpp-sasl' />
    <config-version-mismatch xmlns='urn:xmpp:iap:0' />
    <text>Configuration version mismatch</text>
  </failure>

```

`<!-- Client will then proceed as the previous example -->`

## 4 Security Considerations

Servers which attempt interpretation of matching non-identical tokens can cause clients to miss opportunities to improve their security, or gain new features (which might include security features). So again, this remains NOT RECOMMENDED.

There is a temptation to stuff the entirety of the initial pipelined send into TLS 0-RTT early-data; that is also NOT RECOMMENDED because it may allow for replay attacks. However, where explicit protection is in place for replay (such as FAST) this is acceptable.

## 5 XMPP Registrar Considerations

### 5.1 Protocol Namespaces

- 'urn:xmpp:iap:0'

### 5.2 Stream Features

- 'urn:xmpp:iap:0'

## 6 IANA Considerations

This document requires no interaction with the [Internet Assigned Numbers Authority \(IANA\)](https://www.iana.org/)<sup>3</sup>.

---

<sup>3</sup>The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.