XEP-0192: Proposed Stream Feature Improvements

Peter Saint-Andre
mailto:xsf@stpeter.im
xmpp:peter@jabber.org
http://stpeter.im/

2012-02-08
Version 1.2

Status	Type	Short Name
Obsolete	Standards Track	N/A

This document proposes improvements to the XML stream features definition for inclusion in the specification that supersedes RFC 3920.

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

RFC 3920 introduced the concept of stream features. Implementation experience has revealed several shortcomings in the current definition and usage of stream features:

- Because not all stream features include a mechanism for specifying that negotiation of the feature is required, servers and clients cannot know with certainty when the stream negotiation has been completed and therefore when it is acceptable to begin sending XML stanzas over the stream.
- The server dialback protocol does not have a stream feature associated with it.

Those shortcomings are addressed in this document.
Note: The recommendations from this document were NOT incorporated into RFC 6120 [1] and this document is Obsolete.

2 Required Flag

The XMPP stream feature for Transport Layer Security (TLS) includes a <required/> child element that can be used to indicate that negotiation of TLS must be completed before proceeding with the rest of the stream negotiation. However, as defined in RFC 3920 the remaining stream features do not include the ability to flag that negotiation of the feature is required in order to (1) proceed with the negotiation or (2) begin sending XML stanzas. Because the non-TLS features lack a required flag, it is not possible for the initiating entity to know definitively how to proceed at any given stage in the stream negotiation, and the only way for the initiating entity to know whether it may begin sending XML stanzas is to attempt to send them (the receiving entity will return a <not-authorized/> stream error if not all required features have been negotiated). This state of affairs is suboptimal. Therefore, every stream feature must include the ability to flag the feature as required or not required. When the initiating entity receives a stream features element with no features containing a <required/> element, it knows thatt the receiving party will accept XML stanzas over the stream.
The following examples show a possible flow of stream negotiation between a client and a server, using the required flag for all but one of the features and following the order specified in Recommended Order of Stream Feature Negotiation (XEP-0170) [2]. (This example is more verbose than a typical stream negotiation flow, but is provided here for the sake of completeness.)

[1] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[2] XEP-0170: Recommended Order of Stream Feature Negotiation <https://xmpp.org/extensions/xep-0170.html>.

Listing 1: A stream negotiation

```
C: <stream:stream
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'
     to='example.com'
     version='1.0'>

S: <stream:stream
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'
     id='c2s_123'
     from='example.com'
     version='1.0'>

S: <stream:features>
     <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'>
       <required/>
     </starttls>
   </stream:features>

C: <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>

S: <proceed xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>

[TLS negotiation]

C: <stream:stream
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'
     to='example.com'
     version='1.0'>

S: <stream:stream
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'
     from='example.com'
     id='c2s_234'
     version='1.0'>

S: <stream:features>
     <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
       <mechanism>EXTERNAL</mechanism>
       <mechanism>DIGEST-MD5</mechanism>
       <mechanism>PLAIN</mechanism>
       <required/>
     </mechanisms>
   </stream:features>

C: <auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
```

```
        mechanism='DIGEST-MD5'/>

S: <challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
     cmVhbG09InNvbWVyZWFsbSIsbm9uY2U9Ik9BNk1HOXRFUUdtMmhoIixxb3A9ImF1dGgi

     LGNoYXJzZXQ9dXRmLTgsYWxnb3JpdGhtPW1kNS1zZXNzCg==
   </challenge>

C: <response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
     dXNlcm5hbWU9InNvbWVub2RlIixyZWFsbT0ic29tZXJlYWxtIixub25jZT0i
     T0E2TUc5dEVRR20yaGgiLGNub25jZT0iT0E2TUhYaDZWcVRyUmsiLG5jPTAw
     MDAwMDAxLHFvcD1hdXRoLGRpZ2VzdC11cmk9InhtcHAvZXhhbXBsZS5jb20i
     LHJlc3BvbnNlPWQzODhkYWQ5MGQ0YmJkNzYwYTE1MjMyMWYyMTQzYWY3LGNo
     YXJzZXQ9dXRmLTgK
   </response>

S: <challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
     cnNwYXV0aD1lYTQwZjYwMzM1YzQyN2I1NTI3Yjg0ZGJhYmNkZmZmZAo=
   </challenge>

C: <response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>

S: <success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>

C: <stream:stream
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'
     to='example.com'
     version='1.0'>

S: <stream:stream
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'
     id='c2s_345'
     from='example.com'
     version='1.0'>

S: <stream:features>
     <compression xmlns='http://jabber.org/features/compress'>
       <method>zlib</method>
       <required/>
     </compression>
   </stream:features>

C: <compress xmlns='http://jabber.org/protocol/compress'>
     <method>zlib</method>
   </compress>

S: <compressed xmlns='http://jabber.org/protocol/compress'/>
```

```
C: <stream:stream
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'
     to='example.com'
     version='1.0'>

S: <stream:stream
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'
     id='c2s_456'
     from='example.com'
     version='1.0'>

S: <stream:features>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
       <required/>
     </bind>
   </stream:features>

C: <iq type='set' id='bind_1'>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
       <resource>someresource</resource>
     </bind>
   </iq>

S: <iq type='result' id='bind_1'>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
       <jid>somenode@example.com/someresource</jid>
     </bind>
   </iq>
```

3 Dialback Stream Feature

As specified in RFC 3920, support for the server dialback protocol is currently adverised through inclusion of a dialback namespace prefix in the stream header:

Listing 2: Stream header with dialback namespace advertisement

```
<stream:stream
    xmlns:stream='http://etherx.jabber.org/streams'
    xmlns='jabber:server'
    xmlns:db='jabber:server:dialback'
    id='s2s_123'>
```

However, it is not clear if inclusion of the dialback namespace indicates that a server supports the server dialback protocol or that it requires negotiation of server dialback. To make this

clear, we define a stream feature for server dialback:

Listing 3: Dialback stream feature

```
<stream:features>
  <dialback xmlns='urn:xmpp:features:dialback'>
    <required/>
  </dialback>
</stream:features>
```

Consider the following scenario, in which Server1 provides a self-signed certificate. According to Server2's local service policy, it does not consider self-signed certificates to be trustworthy and therefore requires negotiation of server dialback in this case.

Listing 4: A stream negotiation with server dialback

```
S1: <stream:stream
      xmlns='jabber:server'
      xmlns:db='jabber:server:dialback'
      xmlns:stream='http://etherx.jabber.org/streams'
      to='example.com'
      version='1.0'>

S2: <stream:stream
      xmlns='jabber:server'
      xmlns:db='jabber:server:dialback'
      xmlns:stream='http://etherx.jabber.org/streams'
      id='s2s_123'
      from='example.com'
      version='1.0'>

S2: <stream:features>
      <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'>
        <required/>
      </starttls>
    </stream:features>

S2: <proceed xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>

[TLS negotiation with self-signed certificate]

S1: <stream:stream
      xmlns='jabber:server'
      xmlns:db='jabber:server:dialback'
      xmlns:stream='http://etherx.jabber.org/streams'
      to='example.com'
      version='1.0'>

S2: <stream:stream
```

```
      xmlns='jabber:server'
      xmlns:db='jabber:server:dialback'
      xmlns:stream='http://etherx.jabber.org/streams'
      from='example.com'
      id='c2s_234'
      version='1.0'>

S2: <stream:features>
      <dialback xmlns='urn:xmpp:features:dialback'>
        <required/>
      </dialback>
    </stream:features>

[Dialback negotiation]
```

4 Security Considerations

The improvements described herein do not introduce any new security concerns above and beyond those defined in RFC 3920.

5 IANA Considerations

No interaction with the Internet Assigned Numbers Authority (IANA) [3] is required as a result of this document.

6 XMPP Registrar Considerations

6.1 Stream Features

As specified in Server Dialback (XEP-0220) [4], the XMPP Registrar [5] includes a dialback stream feature of 'urn:xmpp:features:dialback' in its registry of stream features (see <https://xmpp.org/registrar/stream-features.html>).

[3] 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/>.

[4] XEP-0220: Server Dialback <https://xmpp.org/extensions/xep-0220.html>.

[5] The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <https://xmpp.org/registrar/>.

7 XML Schema

7.1 SASL Stream Feature

Note: The following provisional schema is intended to replace the existing schema for the SASL stream feature.

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:ietf:params:xml:ns:xmpp-sasl'
    xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
    elementFormDefault='qualified'>

  <xs:element name='mechanisms'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='mechanism'
                    maxOccurs='unbounded'
                    type='xs:string'/>
        <xs:element name='required'
                    minOccurs='0'
                    maxOccurs='1'
                    type='empty'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='auth'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='xs:string'>
          <xs:attribute name='mechanism'
                        type='xs:string'
                        use='optional'/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name='challenge' type='xs:string'/>
  <xs:element name='response' type='xs:string'/>
  <xs:element name='abort' type='empty'/>
  <xs:element name='success' type='empty'/>

  <xs:element name='failure'>
    <xs:complexType>
      <xs:choice minOccurs='0'>
```

```
        <xs:element name='aborted' type='empty'/>
        <xs:element name='incorrect-encoding' type='empty'/>
        <xs:element name='invalid-authzid' type='empty'/>
        <xs:element name='invalid-mechanism' type='empty'/>
        <xs:element name='mechanism-too-weak' type='empty'/>
        <xs:element name='not-authorized' type='empty'/>
        <xs:element name='temporary-auth-failure' type='empty'/>
      </xs:choice>
    </xs:complexType>
  </xs:element>

  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

7.2 Resource Binding Stream Feature

Note: The following provisional schema is intended to replace the existing schema for the Resource Binding stream feature.

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:ietf:params:xml:ns:xmpp-bind'
    xmlns='urn:ietf:params:xml:ns:xmpp-bind'
    elementFormDefault='qualified'>

  <xs:element name='bind'>
    <xs:complexType>
      <xs:sequence>
        <xs:choice minOccurs='0' maxOccurs='1'>
          <xs:element name='resource' type='resourceType'/>
          <xs:element name='jid' type='fullJIDType'/>
        </xs:choice>
        <xs:element name='required'
                    minOccurs='0'
                    maxOccurs='1'
                    type='empty'/>
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
  <xs:simpleType name='resourceType'>
    <xs:restriction base='xs:string'>
      <xs:minLength value='1'/>
      <xs:maxLength value='1023'/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name='fullJIDType'>
    <xs:restriction base='xs:string'>
      <xs:minLength value='8'/>
      <xs:maxLength value='3071'/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

7.3 Server Dialback Stream Feature

Note: The following defines a schema for the proposed Server Dialback stream feature.

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:features:dialback'
    xmlns='urn:xmpp:features:dialback'
    elementFormDefault='qualified'>

  <xs:element name='dialback'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='required'
                    minOccurs='0'
                    maxOccurs='1'
                    type='empty'/>
      </xs:sequence>
    </xs:complexType>

  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

8 Acknowledgements

Thanks to Ralph Meijer and Joe Hildebrand for their comments.