XEP-0155: Stanza Session Negotiation

Ian Paterson
mailto:ian.paterson@clientside.co.uk
xmpp:ian@zoofy.com

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

2008-01-14
Version 1.2

Status	Type	Short Name
Draft	Standards Track	ssn

This specification defines a method for formally negotiating the exchange of XML stanzas between two XMPP entities. The method uses feature negotiation forms sent via XMPP message stanzas to enable session initiation between entities that do not share presence information or have knowledge of full JabberIDs and therefore is also suitable for use across gateways to SIP-based systems. A wide range of session parameters can be negotiated, including the use of end-to-end encryption, chat state notifications, XHTML-IM formatting, and message archiving.

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 <http://xmpp.org/extensions/ipr-policy.shtml> or obtained by writing to XMPP Standards Foundation, 1899 Wynkoop Street, Suite 600, Denver, CO 80202 USA).

Contents

1 Introduction 1

2 Requirements 1

3 State Chart 2

4 Negotiating a New Session 3
- 4.1 Initiating a Session . . . 3
- 4.2 Accepting a Session . . . 6
- 4.3 Rejecting a Session . . . 7
- 4.4 Completing or Canceling the Negotiation . . . 9

5 Moving A Session to a Different Resource 10

6 Renegotiating a Session 11

7 Terminating a Session 13

8 Defined Parameters 14

9 Implementation Notes 16
- 9.1 Auto Accept or Reject . . . 16
- 9.2 Persisting Sessions . . . 16
- 9.3 Sharing Presence . . . 16
- 9.4 Unavailable Presence . . . 16
- 9.5 Mapping to SIP . . . 17

10 Security Considerations 17
- 10.1 Presence Leaks . . . 17
- 10.2 Localization . . . 18

11 IANA Considerations 18

12 XMPP Registrar Considerations 18
- 12.1 Protocol Namespaces . . . 18
- 12.2 Service Discovery Features . . . 18
- 12.3 Field Standardization . . . 19

13 XML Schema 21

14 Acknowledgements 21

1 Introduction

The traditional model for one-to-one chat ”sessions” in Jabber/XMPP is for a user to simply send a message to a contact with a thread ID but without any formal negotiation of session parameters (see Best Practices for Message Threads [1]). This informal approach to initiation of a session is perfectly acceptable in many contexts, environments, and cultures. However, it may be desirable to formally request a chat session (or any other type of XMPP stanza session) and negotiate its parameters before beginning the session in some circumstances, such as:

- Whenever parameters specific to a stanza session must be agreed. e.g., security and privacy parameters (see Encrypted Session Negotiation [2] and Message Archiving [3]).
- The parties are unknown to each other, have not exchanged presence, or have not discovered their respective capabilities via Service Discovery [4] or Entity Capabilities [5].
- When an XMPP-based system interfaces with a SIP-based system built on top of RFC 3261 [6]. [7]
- Within an organization or culture in which one would not simply begin interacting with another person (e.g., a superior) without first receiving permission to do so.

This proposal defines best practices for such a negotiation, re-using the protocol defined in Feature Negotiation [8].

2 Requirements

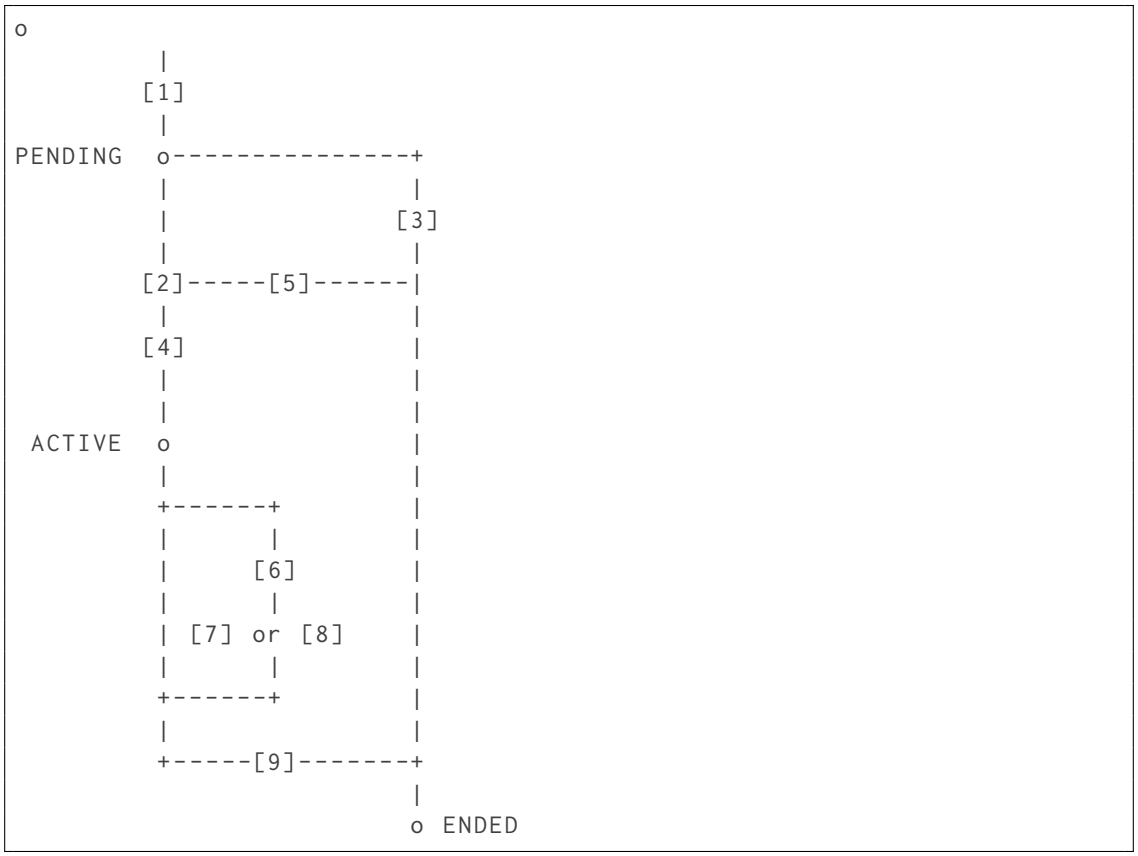
The specification addresses the following use cases:

- Negotiating a new stanza session
- Moving an existing stanza session from one resource to another
- Renegotiating an existing stanza session
- Terminating an existing stanza session

[1] XEP-0201: Best Practices for Message Threads <http://xmpp.org/extensions/xep-0201.html>.
[2] XEP-0116: Encrypted Session Negotiation <http://xmpp.org/extensions/xep-0116.html>.
[3] XEP-0136: Message Archiving <http://xmpp.org/extensions/xep-0136.html>.
[4] XEP-0030: Service Discovery <http://xmpp.org/extensions/xep-0030.html>.
[5] XEP-0115: Entity Capabilities <http://xmpp.org/extensions/xep-0115.html>.
[6] RFC 3261: Session Initiation Protocol (SIP) <http://tools.ietf.org/html/rfc3261>.
[7] In essence, a stanza session negotiation request as specified herein is functionally equivalent to a SIP INVITE request, and acceptance of such a request is functionally equivalent to sending a SIP 200 OK response; see Section 17 of RFC 3261.
[8] XEP-0020: Feature Negotiation <http://xmpp.org/extensions/xep-0020.html>.

3 State Chart

The following figure attempts to capture the state transitions in visual form.



[1] A stanza session negotiation is initiated when the user sends a message containing a data form of type ”form” with an ”accept” field.

A stanza session negotiation is accepted when the contact sends a message containing a data form of type ”submit” with an ”accept” field whose value is ”1” or ”true”.

A stanza session negotiation is rejected when the contact sends a message containing a data form of type ”submit” with an ”accept” field whose value is ”0” or ”false”.

A stanza session negotiation is completed when the user sends a message containing a data form of type ”result” with an ”accept” field whose value is ”1” or ”true”.

A stanza session negotiation is canceled when the user sends a message containing a data form of type ”result” with an ”accept” field whose value is ”0” or ”false”.

An existing session is re-negotiated when either party sends a message containing a data form of type ”form” with a ”renegotiate” field whose value is ”1” or ”true”.

A session re-negotiation is accepted when the other party sends a message containing a data form of type ”submit” with a ”renegotiate” field whose value is ”1” or ”true”.

A session re-negotiation is rejected when the other party sends a message containing a data form of type ”submit” with a ”renegotiate” field whose value is ”0” or ”false”; however, the session remains in the active state with the previously-negotiated parameters in force.

A session is terminated when either party sends a message containing a data form of type ”submit” with a ”terminate” field whose value is ”1” or ”true”.

4 Negotiating a New Session

4.1 Initiating a Session

In order to initiate a negotiated session, the initiating party (”user”) sends a <message/> [9] stanza to the receiving party (”contact”) containing a <feature/> child qualified by the ’http://jabber.org/protocol/feature-neg’ namespace. The <message/> stanza MUST NOT contain a <body/> child element (as specified in RFC 3921 [10]). The <message/> stanza type SHOULD be ”normal” (either explicitly or by non-inclusion of the ’type’ attribute). The stanza MUST contain a <thread/> element for tracking purposes (where the newly-generated ThreadID is unique to the proposed session). The data form MUST contain a hidden FORM_TYPE field whose value is ”urn:xmpp:ssn” and MUST contain a boolean field named ”accept”. [11] The inclusion of ”logging”, ”disclosure” and ”security” fields is also RECOMMENDED. Note: The options within any ’list-single’ fields SHOULD appear in order of preference.
Note: Sessions may be conducted between entities who are never online at the same time. However, if the user is interested only in an immediate session then the user SHOULD instruct the contact’s server not to store the message for later delivery (see Best Practices for Handling Offline Messages [12]) using the Advanced Message Processing [13] protocol.
In the following example of a negotiation request, Romeo requests a chat with Juliet and also queries her regarding whether she is able to disallow all message logging (see Message Archiving [14]) [15], whether she wants to temporarily share presence for this session (see the Sharing Presence section of this document), and whether she wants to support the XHTML-IM [16] and Chat State Notifications [17] extensions during this session. He asks Juliet’s client if it is prepared to make a (legally binding) guarantee that it does not intentionally implement

[9] The <message/> stanza is used because the user does not necessarily know which of the contact’s resources is most available (or indeed if the contact is online).
[10] RFC 3921: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc3921>.
[11] In accordance with Section 3.2.2.1 of XML Schema Part 2: Datatypes, the allowable lexical representations for the xs:boolean datatype are the strings ”0” and ”false” for the concept ’false’ and the strings ”1” and ”true” for the concept ’true’; implementations MUST support both styles of lexical representation.
[12] XEP-0160: Best Practices for Handling Offline Messages <http://xmpp.org/extensions/xep-0160.html>.
[13] XEP-0079: Advanced Message Processing <http://xmpp.org/extensions/xep-0079.html>.
[14] XEP-0136: Message Archiving <http://xmpp.org/extensions/xep-0136.html>.
[15] A client MUST NOT set the ’logging’ field to ’mustnot’ unless it has confirmed that its server will allow it to switch off Automated Archiving (see Message Archiving).
[16] XEP-0071: XHTML-IM <http://xmpp.org/extensions/xep-0071.html>.
[17] XEP-0085: Chat State Notifications <http://xmpp.org/extensions/xep-0085.html>.

any feature (not even a disabled feature) that might disclose the content of the session, any associated (decryption) keys, or his identity to any third-party (see Encrypted Session Negotiation). He also requires that they are both connected securely to their servers, and asks which language she prefers amongst those he can write.
Note: These fields are examples only. For definitions of these fields, refer to the Defined Parameters section of this document.

Listing 1: User requests session

```
<message type='normal'
         from='romeo@montague.net/orchard'
         to='juliet@capulet.com'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='form'>
      <title>Open chat with Romeo?</title>
      <field var='FORM_TYPE' type='hidden'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field label='Accept this session?' type='boolean' var='accept'>
        <value>true</value>
        <required/>
      </field>
      <field label='Message logging' type='list-single' var='logging'>
        <value>mustnot</value>
        <option label='Allow message logging'>
          <value>may</value>
        </option>
        <option label='Disallow all message logging'>
          <value>mustnot</value>
        </option>
        <required/>
      </field>
      <field label="Disclosure" type="list-single" var="disclosure">
        <value>never</value>
        <option label="Guarantee disclosure not implemented">
          <value>never</value>
        </option>
        <option label="Disable all disclosures">
          <value>disabled</value>
        </option>
        <option label="Allow disclosures">
          <value>enabled</value>
        </option>
        <required/>
      </field>
      <field label='Allow multiple sessions?' type='boolean' var='
          multisession'>
        <value>false</value>
```

```
      </field>
      <field label='XHTML formatting'
             type='list-single'
             var='http://jabber.org/protocol/xhtml-im'>
        <value>may</value>
        <option label='Allow XHTML formatting'><value>may</value></
            option>
        <option label='Disallow XHTML formatting'><value>mustnot</
            value></option>
      </field>
      <field label='Temporarily share presence?'
             type='list-single'
             var='presence'>
        <value>may</value>
        <option label='Allow temporary presence sharing'><value>may</
            value></option>
        <option label='Disallow temporary presence sharing'><value>
            mustnot</value></option>
      </field>
      <field label='Chat State Notifications'
             type='list-single'
             var='http://jabber.org/protocol/chatstates'>
        <value>may</value>
        <option label='Allow Chat State Notifications'><value>may</
            value></option>
        <option label='Disallow Chat State Notifications'><value>
            mustnot</value></option>
      </field>
      <field label='Minimum security level'
             type='list-single'
             var='security'>
        <value>c2s</value>
        <option label='Both parties must be securely connected to
            their servers'>
          <value>c2s</value>
        </option>
        <required/>
      </field>
      <field label='Primary written language of the chat'
             type='list-single'
             var='language'>
        <value>en</value>
        <option label='English'><value>en</value></option>
        <option label='Italiano'><value>it</value></option>
      </field>
    </x>
  </feature>
  <amp xmlns='http://jabber.org/protocol/amp'>
    <rule action='drop' condition='deliver' value='stored'/>
```

```
  </amp>
</message>
```

The user MAY request a session with a specific resource of the contact. However, if the user specifies no resource (or if the specified resource is not available), then the contact's server delivers the request to the contact's most available resource (which in the examples below happens to be "balcony"). If no resource is available (and no Advanced Message Processing rule included in the request specifies otherwise) then the server MAY store the request for later delivery.

4.2 Accepting a Session

If, upon reception of a user's session request, a contact finds that the request had been stored for later delivery, and if the contact is interested only in an immediate session, then it SHOULD initiate a new stanza session negotiation (including a newly-generated ThreadID) instead of responding to the user's request. Note: Sending any response to the user's original request would leak presence information since it would divulge the fact that the contact had been offline rather than just ignoring the user.
In any response to the user's request, the contact's client MUST mirror the <thread/> value so that the user's client can correctly track the response. The <message/> stanza MUST NOT contain a <body/> child element.
If the request is accepted then the contact's client MUST include in its response values for all the fields that the request indicated are required. If the contact's client does not support one of the default values or if the contact has disabled its support (as for Chat State Notifications and XHTML formatting in the example below), and the client can still accept the request, then it MUST set that field to a value that it can support.
In the example below we assume that Juliet accepts the session and specifies that she prefers to speak Italian with Romeo:

Listing 2: Contact accepts session and specifies parameters

```
<message  type='normal'
          from='juliet@capulet.com/balcony'
          to='romeo@montague.net/orchard'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature  xmlns='http://jabber.org/protocol/feature-neg'>
    <x  xmlns='jabber:x:data'  type='submit'>
      <field  var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field  var='accept'><value>true</value></field>
      <field  var='logging'><value>mustnot</value></field>
      <field  var='disclosure'><value>never</value></field>
      <field  var='http://jabber.org/protocol/xhtml-im'>
        <value>may</value>
```

```
      </field>
      <field var='http://jabber.org/protocol/chatstates'>
        <value>may</value>
      </field>
      <field var='security'><value>c2s</value></field>
      <field var='language'><value>it</value></field>
    </x>
  </feature>
</message>
```

Note: Both entities MUST assume the session is being established with the resource of the contact that sends the reply, even if the user sent its request to a different resource of the contact.

4.3 Rejecting a Session

If the contact does not want to reveal presence to the user for whatever reason then the contact's client SHOULD return no response or error (see Presence Leaks). Also, if the contact is using a legacy client then it MAY not support returning any response or error. In both these cases the user MAY proceed to send stanzas to the contact outside the context of a negotiated session.
However, if the contact simply prefers not to start a session then the client SHOULD decline the invitation. The data form MUST contain the FORM_TYPE field and the "accept" field set to "0" or "false". It is RECOMMENDED that the form does not contain any other fields even if the request indicated they are required. The client MAY include a reason via the "reason" field (which is of type "text-single"). The <message/> stanza MUST NOT contain a <body/> child element.

Listing 3: Contact declines offer and specifies reason

```
<message type='normal'
         from='juliet@capulet.com/balcony'
         to='romeo@montague.net/orchard'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='accept'><value>0</value></field>
      <field var='reason'>
        <value>Sorry, can't chat now! How about tonight?</value>
      </field>
    </x>
  </feature>
</message>
```

If the contact's client does not support feature negotiation or does not support the "urn:xmpp:ssn" FORM_TYPE, it SHOULD return a <service-unavailable/> error:

Listing 4: Contact returns service unavailable error

```
<message type='error'
         from='juliet@capulet.com/balcony'
         to='romeo@montague.net/orchard'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='form'>
      <field var='FORM_TYPE' type='hidden'>
        <value>urn:xmpp:ssn</value>
      </field>
      ...
    </x>
  </feature>
  <error code='503' type='cancel'>
    <service-unavailable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</message>
```

If the contact's client does not support one or more of the required features, it SHOULD return a <feature-not-implemented/> error, specifying the field(s) not implemented using the 'var' attribute of one or more <field/> child elements of a <feature/> child element of the <error/> scoped by the 'http://jabber.org/protocol/feature-neg' namespace:

Listing 5: Contact returns feature not implemented error

```
<message type='error'
         from='juliet@capulet.com/balcony'
         to='romeo@montague.net/orchard'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='form'>
      <field var='FORM_TYPE' type='hidden'>
        <value>urn:xmpp:ssn</value>
      </field>
      ...
    </x>
  </feature>
  <error code='501' type='cancel'>
    <feature-not-implemented xmlns='urn:ietf:params:xml:ns:xmpp-
       stanzas'/>
    <feature xmlns='http://jabber.org/protocol/feature-neg'>
      <field var='logging'/>
    </feature>
  </error>
</message>
```

If the contact's client supports none of the options for one or more required fields, it SHOULD return a <not-acceptable/> error, specifying the field(s) with unsupported options using the 'var' attribute of one or more <field/> child elements of a <feature/> child element of the <error/> scoped by the 'http://jabber.org/protocol/feature-neg' namespace:

Listing 6: Contact returns options not acceptable error

```
<message type='error'
         from='juliet@capulet.com/balcony'
         to='romeo@montague.net/orchard'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='form'>
      <field var='FORM_TYPE' type='hidden'>
        <value>urn:xmpp:ssn</value>
      </field>
      ...
    </x>
  </feature>
  <error code='406' type='modify'>
    <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <feature xmlns='http://jabber.org/protocol/feature-neg'>
      <field var='security'/>
    </feature>
  </error>
</message>
```

4.4 Completing or Canceling the Negotiation

If the contact accepted the session (see Accepting a Session) then the user MUST either complete or cancel the stanza session negotiation. If the contact chose an option other than the default (prefered) value for one or more of the fields, then instead of having the client accept the session automatically the user may prefer to review the values that the contact selected before confirming that the session is open. [18] In any case the user's client SHOULD verify that the selected values are acceptable before completing the stanza session negotiation -- and confirming that the session is open -- by replying with a form with the form 'type' attribute set to 'result'. The form MUST contain the FORM_TYPE field and the "accept" field set to "1" or "true". The user MAY include an explanation or reason via the "reason" field (which is of type "text-single"). The <message/> stanza MUST NOT contain a <body/> child element.

Listing 7: User completes negotiation and confirms session is open

[18] See Encrypted Session Negotiation for example of other instances where the user might find the values submitted by the contact unacceptable.

```
<message type='normal'
         from='romeo@montague.net/orchard'
         to='juliet@capulet.com/balcony'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='result'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='accept'><value>true</value></field>
      <field var='reason'>
        <value>I forgot what I wanted to say!</value>
      </field>
    </x>
  </feature>
</message>
```

Alternatively, if the user decides to cancel the stanza session negotiation then the client MUST reply with a data form containing the FORM_TYPE field and the "accept" field set to "0" or "false":

Listing 8: User cancels stanza session negotiation

```
<message type='normal'
         from='romeo@montague.net/orchard'
         to='juliet@capulet.com/balcony'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='result'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='accept'><value>0</value></field>
    </x>
  </feature>
</message>
```

5 Moving A Session to a Different Resource

Either party MAY ask to continue the session using another of its resources. The requesting party does this by submitting a form with a "continue" field containing the value of the new resource:

Listing 9: One party asks to switch session to another of its resources

```
<message type='normal'
         from='juliet@capulet.com/balcony'
```

```
        to='romeo@montague.net/orchard'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='continue'><value>PDA</value></field>
    </x>
  </feature>
</message>
```

The requesting party SHOULD NOT send stanzas within the session from either resource until the other party has accepted the switch to the new resource.
The other client SHOULD accept the switch automatically since the requesting party might otherwise be unable to continue the session:

Listing 10: Other client accepts switch

```
<message type='normal'
         from='romeo@montague.net/orchard'
         to='juliet@capulet.com/balcony'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='result'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='continue'><value>PDA</value></field>
    </x>
  </feature>
</message>
```

Once the other party has accepted the switch then all stanzas sent within the session MUST be to or from the new resource. Note: Both parties MUST ensure that they comply with all the other stanza session negotiation parameters that were previously agreed for this session.

6 Renegotiating a Session

At any time during an existing session, either party MAY attempt to renegotiate the parameters of the session using the protocol described in Negotiating a New Session. The requesting party does this by sending a new <message/> stanza containing a feature negotiation form and a <thread/> element with the same value as that of the existing session. Note: The "accept" field MUST NOT be included in a renegotiation form and the <message/> stanza MUST NOT contain a <body/> child element. The other fields MAY be different from the set of fields

included in the initial stanza session negotiation form.

Listing 11: One party requests renegotiation

```
<message type='normal'
         from='juliet@capulet.com/balcony'
         to='romeo@montague.net/orchard'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='form'>
      <field var='FORM_TYPE' type='hidden'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field label='Renegotiate?' type='boolean' var='renegotiate'>
        <value>1</value>
        <required/>
      </field>
      <field label='Message logging' type='list-single' var='logging'>
        <value>mustnot</value>
        <option label='Disallow all message logging'>
          <value>may</value>
        </option>
        <required/>
      </field>
    </x>
  </feature>
</message>
```

The requesting party MAY continue to send stanzas within the session while it is waiting for the other party to either accept the parameters or report an error.
In order to accept the renegotiation, the other party shall send a message containing a data form of type "submit" with the 'renegotiate' field set to a value of "1" or "true".

Listing 12: Other party accepts renegotiation and specifies parameters

```
<message type='normal'
         from='romeo@montague.net/orchard'
         to='juliet@capulet.com/balcony'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='renegotiate'><value>1</value></field>
      <field var='logging'><value>may</value></field>
    </x>
  </feature>
</message>
```

Note: Both parties MUST consider the renegotiation to be complete as soon as the parameter acceptance message has been sent (or received).
Note: The requesting party SHOULD NOT send a renegotiation completion or cancelation message (see Completing or Canceling the Negotiation).
Note: Both parties MUST ensure that they continue to comply with all the stanza session negotiation parameters that were not renegotiated but had previously been agreed for this session.
In order to reject the renegotiation, the other party shall send a message containing a data form of type "submit" with the 'renegotiate' field set to a value of "0" or "false".

Listing 13: Other party rejects renegotiation

```
<message type='normal'
         from='romeo@montague.net/orchard'
         to='juliet@capulet.com/balcony'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='renegotiate'><value>0</value></field>
      <field var='logging'><value>may</value></field>
    </x>
  </feature>
</message>
```

If the other party's client does not support one or more of the required features, it SHOULD return a <feature-not-implemented/> error. If the other party's client supports none of the options for one or more required fields, it SHOULD return a <not-acceptable/> error (see Rejecting a Session). Note: In any of these cases the existing negotiated session parameters are maintained. Either party MAY choose to terminate the session only as specified in the section Terminating a Session.

7 Terminating a Session

In order to explicitly terminate a negotiated session, the party that wishes to end the session MUST do so by sending a <message/> containing a data form of type "submit". The <message/> stanza MUST contain a <thread/> element with the same XML character data as the original initiation request. The <message/> stanza MUST NOT contain a <body/> child element. The data form containing a boolean field named "terminate" set to a value of "1" or "true".

Listing 14: One party terminates session

```
<message type='normal'
         from='juliet@capulet.com/balcony'
         to='romeo@montague.net/orchard'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='terminate'><value>1</value></field>
    </x>
  </feature>
</message>
```

Both parties MUST then consider the session to be ended.
The other party's client MAY explicitly acknowledge the termination of the session by sending a <message/> containing a data form of type "result", and the value of the "terminate" field set to "1" or "true" (see Encrypted Session Negotiation for a practical example). The client MUST mirror the <thread/> value it received.

Listing 15: Other party acknowledges session termination

```
<message type='normal'
         from='romeo@montague.net/orchard'
         to='juliet@capulet.com/balcony'>
  <thread>ffd7076498744578d10edabfe7f4a866</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='result'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='terminate'><value>1</value></field>
    </x>
  </feature>
</message>
```

8 Defined Parameters

This section defines the parameters for stanza session negotiation parameters and whether they must, should, or may be included in the initial negotiation form. Additional parameters may be registered as described in the XMPP Registrar Considerations section of this document.

Name
accept
continue
disclosure

9 Implementation Notes

9.1 Auto Accept or Reject

A client MAY require a human user to approve each stanza session negotiation request, however it is RECOMMENDED that it accepts or rejects automatically as many requests as possible, based on a set of user-configurable policies (see Presence Leaks).

9.2 Persisting Sessions

Stanza session negotiation sometimes requires the involvement of either or both human users, and if human input is required but the user is away then session establishment may be delayed indefinitely. So, in order to minimise the number of user interruptions and delays, clients SHOULD reuse existing sessions whenever possible. For example, a client SHOULD NOT terminate sessions unless the user is going offline, even if its user closes a window associated with the session.

9.3 Sharing Presence

If so negotiated via the 'presence' field, two parties who do not have subscriptions to each other's presence (as specified in XMPP-IM) may share presence by sending directed presence after the session is negotiated.

Listing 16: User sends directed presence to contact

```
<presence from='romeo@montague.net/orchard' to='juliet@capulet.com/
    balcony'/>
```

Listing 17: Contact sends directed presence to user

```
<presence from='juliet@capulet.com/balcony' to='romeo@montague.net/
    orchard'/>
```

In accordance with the rules specified in XMPP-IM, sharing presence enables one party's server to send unavailable presence to the other party if the sending party goes offline for any reason.

9.4 Unavailable Presence

If a party receives an XMPP presence stanza of type "unavailable" from the full JID <local-part@domain.tld/resource> of the other party (i.e., the resource with which it has had an active session) during a session, the receiving party SHOULD assume that the other client will still be able to continue the session (perhaps it simply became "invisible", or it is persisting the state of the negotiated session until it reconnects and receives "offline" messages).

However, the receiving party MAY assume that the other client will not be able to continue the session. [19] In that case it MUST explicitly terminate the session (see Terminating a Session) -- since its assumption could be incorrect. If after terminating the session the receiving party later receives available presence (i.e., a <presence/> stanza with no 'type' attribute) from that same resource or another resource associated with the other party and the receiving party desires to restart the session, then it MUST initiate a new session (including a newly-generated ThreadID) with the other party. It MUST NOT renegotiate parameters for the terminated session. (Note: This is consistent with the handling of chat states as specified in XEP-0085.)

9.5 Mapping to SIP

When mapping instant messaging flows to SIP, implementations SHOULD adhere to draft-saintandre-sip-xmpp-im [20].
In addition, the following mappings apply to chat session negotiation:

- Initiation of a negotiated chat session maps to the semantics of the SIP INVITE method.
- Renegotiation of a negotiated chat session also maps to the semantics of the SIP INVITE method.
- Termination of a negotiated chat session maps to the semantics of the SIP BYE method.
- The XMPP <thread/> value maps to the semantics of the SIP Call-ID attribute.

10 Security Considerations

10.1 Presence Leaks

If a contact does not share its presence information with a user through a presence subscription (see RFC 3921) or if it blocks outbound presence notifications to the user (see Privacy Lists [21]), then it will effectively expose its presence if it accepts the user's stanza session negotiation request or returns an error to the user. Therefore, due care must be exercised in determining whether to accept the request or return an error. The contact's client SHOULD NOT automatically (i.e. without first asking the contact) either accept the user's request or return an error to the user unless the user is subscribed to the contact's presence and the contact is not blocking outbound presence notifications to the user. Note: There should be no need for the contact's client to consult the contact's block list (see Simple Communications

[19] In general, if a party is not subscribing to the other party's presence then it will never assume the other party is is unable to continue a session.
[20] Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): Instant Messaging <http://tools.ietf.org/html/draft-saintandre-sip-xmpp-im> (work in progress).
[21] XEP-0016: Privacy Lists <http://xmpp.org/extensions/xep-0016.html>.

Blocking [22]), since if the user is on the block list then the contact would not receive the request from the user in the first place.

10.2 Localization

If a client is configured to show a request <form/> to a human user instead of responding automatically, it SHOULD replace the content of the <title/> element and of all label attributes of the known and registered <field/> and <option/> elements with its own localised versions before showing the form to the user -- even if the form already appears to be in the correct language.
Note: If a client fails to localize the form, a malicious contact might, for example, either switch the labels on the 'security' and 'logging' fields, or use the <title/> to mislead the user regarding the identity of the contact.

11 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [23].

12 XMPP Registrar Considerations

12.1 Protocol Namespaces

The XMPP Registrar [24] includes 'urn:xmpp:ssn' in its registry of protocol namespaces (see <http://xmpp.org/registrar/namespaces.html>).

12.2 Service Discovery Features

The XMPP Registrar includes 'urn:xmpp:ssn' in its registry of Service Discovery features (see <http://xmpp.org/registrar/disco-features.html>).

```
<var>
  <name>urn:xmpp:ssn</name>
```

[22] XEP-0191: Simple Communications Blocking <http://xmpp.org/extensions/xep-0191.html>.
[23] 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/>.
[24] 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 <http://xmpp.org/registrar/>.

```
  <desc>Support for Stanza Session Negotiation and its FORM_TYPE</desc
     >
  <doc>XEP-0155</doc>
</var>
```

12.3 Field Standardization

Field Standardization for Data Forms [25] defines a process for standardizing the fields used within Data Forms qualified by a particular namespace. The following fields are registered for use in Stanza Session Negotiation (see <http://xmpp.org/registrar/formtypes.html>):

```
<form_type>
  <name>urn:xmpp:ssn</name>
  <doc>XEP-0155</doc>
  <desc>
    Forms enabling negotation of a one-to-one
    session between two entities.
  </desc>
  <field
      var='accept'
      type='boolean'
      label='Whether to accept the invitation'/>
  <field
      var='continue'
      type='text-single'
      label='Another resource with which to continue the session'/>
  <field
      var="disclosure"
      type="list-single"
      label="Disclosure of content, decryption keys or identities">
    <option label="Entities guarantee no disclosure features
                   exist (not even disabled features)">
      <value>never</value>
    </option>
    <option label="Entities MUST NOT disclose (except for those
                   disclosures that are required by law)">
      <value>disabled</value>
    </option>
    <option label="Entities MAY disclose">
      <value>enabled</value>
    </option>
  </field>
  <field
      var='http://jabber.org/protocol/chatstates'
      type='list-single'
      label='Whether may send Chat State Notifications per XEP-0085'>
```

[25] XEP-0068: Field Data Standardization for Data Forms <http://xmpp.org/extensions/xep-0068.html>.

```
    <option label='May Send'>
      <value>may</value>
    </option>
    <option label='Must Not Send'>
      <value>mustnot</value>
    </option>
  </field>
  <field
      var='http://jabber.org/protocol/xhtml-im'
      type='list-single'
      label='Whether allowed to use XHTML-IM formatting per XEP-0071'>
    <option label='May Send'>
      <value>may</value>
    </option>
    <option label='Must Not Send'>
      <value>mustnot</value>
    </option>
  </field>
  <field
      var='language'
      type='list-single'
      label='Primary written language of the chat (each
             value appears in order of preference and
             conforms to RFC 4646 and the IANA registry)'/>
  <field
      var='logging'
      type='list-single'
      label='Whether allowed to log messages (i.e.,
             whether Off-The-Record mode is required)'>
    <option label='Allow Message Logging'>
      <value>may</value>
    </option>
    <option label='Disallow All Message Logging (i.e., must
                   disable absolutely all message
                   logging including automatic archiving
                   -- see XEP-0136'>
      <value>mustnot</value>
    </option>
  </field>
  <field
      var='multisession'
      type='boolean'
      label='Whether to allow multiple concurrent sessions between the
          parties'/>
  <field
      var='renegotiate'
      type='boolean'
      label='Whether to renegotiate the session'/>
  <field
```

```
      var='security'
      type='list-single'
      label='Minimum security level'>
    <option label='Secure connections not required'>
      <value>none</value>
    </option>
    <option label='Both parties must be securely connected to their
       servers'>
      <value>c2s</value>
    </option>
    <option label='Both parties must be securely connected to each
       other'>
      <value>e2e</value>
    </option>
  </field>
  <field
      var='terminate'
      type='boolean'
      label='Whether to terminate the session'/>
</form_type>
```

13 XML Schema

This proposal re-uses the format defined in XEP-0020 and therefore does not require a dedicated schema.

14 Acknowledgements

Thanks to Thomas Charron and Jean-Louis Seguineau for their feedback.