XEP-0020: Feature Negotiation

Peter Millard

Peter Saint-Andre
mailto:peter@andyet.net
xmpp:stpeter@stpeter.im
https://stpeter.im/

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

2006-11-21
Version 1.5

Status	Type	Short Name
Draft	Standards Track	feature-neg

This specification defines an XMPP protocol extension that enables two entities to mutually negotiate feature options, such as parameters related to a file transfer or a communications session.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

A discovery protocol such as Service Discovery (XEP-0030) [1] enables Jabber entities to query other entities regarding the features they support, but does not provide a means for the two entities to negotiate specific options related to the advertised features.
The protocol defined herein enables Jabber entities to negotiate options for specific features. These features could be negotiated between any two endpoints on the Jabber network, such as two clients, a client and a component, two components, a client and a server, or two servers. The protocol is generic enough that it can be used whenever options need to be negotiated between two Jabber entities. For examples, Stream Initiation (XEP-0095) [2], SI File Transfer (XEP-0096) [3] or Stanza Session Negotiation (XEP-0155) [4].

2 Protocol Details

Features are negotiated through the exchange of <iq/> or <message/> stanzas containing <feature/> child elements qualified by the 'http://jabber.org/protocol/feature-neg' namespace. However, this <feature/> element is simply a wrapper for structured data encapsulated in the Data Forms (XEP-0004) [5] protocol. [6]
In order to begin a negotation, the initiator sends an <iq/> stanza of type "get" (or a <message/> stanza type "normal" - see Stanza Session Negotiation for examples) to the recipient with a single <feature/> element containing a data form of type "form" which defines the available options for one or more features. Each feature is represented as an x-data "field".
The recipient SHOULD examine each feature and the values of the options provided. In order to indicate preferred values, the recipient then SHOULD specify one value for each feature and return a data form of type "submit" to the initiator in an <iq/> stanza of type "result" (or a <message/> stanza type "normal").
The following examples show some likely scenarios for feature negotiation between entities. Further examples can be found in "using protocols", such as File Transfer.

2.1 Basic Flow

A typical negotiation flow is shown in the following example of two entities negotiating the time and place for a meeting.

[1] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[2] XEP-0095: Stream Initiation <https://xmpp.org/extensions/xep-0095.html>.
[3] XEP-0096: SI File Transfer <https://xmpp.org/extensions/xep-0096.html>.
[4] XEP-0155: Stanza Session Negotiation <https://xmpp.org/extensions/xep-0155.html>.
[5] XEP-0004: Data Forms <https://xmpp.org/extensions/xep-0004.html>.
[6] Earlier versions of this document defined a structured data format to handle the feature negotiation workflow; versions later than 0.4 use Data Forms, i.e., the 'jabber:x:data' namespace.

Listing 1: Initiating entity sends offer

```
<iq type='set'
    from='romeo@montague.net/orchard'
    to='juliet@capulet.com/balcony'
    id='neg1'>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='form'>
      <field var='FORM_TYPE' type='hidden'>
        <value>romantic_meetings</value>
      </field>
      <field type='list-single' var='places-to-meet'>
         <option><value>Secret Grotto</value></option>
         <option><value>Verona Park</value></option>
      </field>
      <field type='list-single' var='times-to-meet'>
         <option><value>22:00</value></option>
         <option><value>22:30</value></option>
         <option><value>23:00</value></option>
      </field>
    </x>
  </feature>
</iq>
```

Listing 2: Responding entity sends preferred option values

```
<iq type='result'
    id='neg1'
    from='juliet@jabber.org/balcony'
    to='romeo@montague.net/orchard'>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='FORM_TYPE'>
        <value>romantic_meetings</value>
      </field>
      <field var='places-to-meet'>
        <value>Secret Grotto</value>
      </field>
      <field var='times-to-meet'>
        <value>22:30</value>
      </field>
    </x>
  </feature>
</iq>
```

Note: If the responding entity does not want to reveal presence to the initiating entity for whatever reason then the responding entity's client SHOULD return a <service-unavailable/> error (or return no response or error whatsoever if the offer was wrapped in a <message/> stanza) - see Security Considerations.

If the responding entity does not support **Feature Negotiation** or does not support the

specified FORM_TYPE, it SHOULD also return a <service-unavailable/> error:

Listing 3: Responding entity does not support feature negotiation

```
<iq type='error'
    id='neg1'
    from='juliet@jabber.org/balcony'
    to='romeo@montague.net/orchard'>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='form'>
      <field var='FORM_TYPE' type='hidden'>
        <value>romantic_meetings</value>
      </field>
      ...
    </x>
  </feature>
  <error type='cancel'>
    <service-unavailable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the responding entity does not support one or more of the features, it SHOULD return a <feature-not-implemented/> error, and SHOULD specify the feature(s) not implemented in the XMPP <text/> element.

Listing 4: Responding entity does not support a feature

```
<iq type='error'
    id='neg1'
    from='juliet@jabber.org/balcony'
    to='romeo@montague.net/orchard'>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='form'>
      <field var='FORM_TYPE' type='hidden'>
        <value>romantic_meetings</value>
      </field>
      ...
    </x>
  </feature>
  <error type='cancel'>
    <feature-not-implemented xmlns='urn:ietf:params:xml:ns:xmpp-
        stanzas'/>
    <text xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>times-to-meet</
        text>
  </error>
</iq>
```

If the responding entity supports none of the options offered for one or more of the features, it SHOULD return a <not-acceptable/> error, and SHOULD specify the relevant feature(s) in

the XMPP <text/> element.

Listing 5: Responding entity supports no options

```
<iq type='error'
    from='juliet@jabber.org/balcony'
    to='romeo@montague.net/orchard'
    id='neg1'>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='form'>
      <field var='FORM_TYPE' type='hidden'>
        <value>romantic_meetings</value>
      </field>
      ...
    </x>
  </feature>
  <error type='modify'>
    <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <text xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>places-to-meet</
       text>
  </error>
</iq>
```

2.2 Querying for Negotiable Features

If at least one feature offered by an entity is subject to **Feature Negotiation**, the entity's response to a service discovery information request MUST include <feature var='http://jabber.org/protocol/feature-neg'/> as one of the features.

Listing 6: Client queries a chatroom for supported features

```
<iq type='get'
    from='juliet@capulet.com/balcony'
    to='balconyscene@plays.shakespeare.lit'
    id='neg1'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 7: Chatroom returns supported features

```
<iq type='result'
    from='balconyscene@plays.shakespeare.lit'
    to='juliet@capulet.com/balcony'
    id='neg1'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    ...
    <feature var='http://jabber.org/protocol/feature-neg'/>
    <feature var='muc-password'/>
```

```
    ...
  </query>
</iq>
```

The "using protocol" (in these examples, Multi-User Chat (XEP-0045) [7]) SHOULD specify which features might be negotiable, either in the relevant documentation or in the entry for that feature in the service discovery features registry maintained by the XMPP Registrar [8]. However, the initiating entity MAY also query the responding entity in order to determine which features are negotiable, as shown below.

Listing 8: Client queries chatroom regarding options for a negotiable feature

```
<iq type='get'
    from='juliet@capulet.com/balcony'
    to='balconyscene@plays.shakespeare.lit'
    id='neg2'>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='muc-password'/>
    </x>
  </feature>
</iq>
```

If that feature is not negotiable, the responding entity SHOULD return a "Feature Not Implemented" error:

Listing 9: Chatroom returns error

```
<iq type='result'
    from='balconyscene@plays.shakespeare.lit'
    to='juliet@capulet.com/balcony'
    id='neg2'>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='muc-password'/>
    </x>
  </feature>
  <error type='cancel'>
    <feature-not-implemented xmlns='urn:ietf:params:xml:ns:xmpp-
        stanzas'/>
  </error>
</iq>
```

[7] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.

[8] 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/>.

If that feature is negotiable, the responding entity SHOULD return an appropriate negotiation form:

Listing 10: Chatroom returns negotiation form

```
<iq type='result'
    from='balconyscene@plays.shakespeare.lit'
    to='juliet@capulet.com/balcony'
    id='neg2'>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='result'>
      <field var='FORM_TYPE'>
        <value>MUC</value>
      </field>
      <field var='muc-password' type='list-single'>
        <option><value>cleartext</value></option>
        <option><value>SHA1</value></option>
        <option><value>SASL</value></option>
      </field>
    </x>
  </feature>
</iq>
```

The initiating entity MAY then submit a data form containing the required information.

3 Security Considerations

If the responding entity responds to the initiating entity or returns an error (other than a <service-unavailable/> response to an <iq/> request), the initiating entity will effectively discover the presence of the responding entity's resource. Due care must therefore be exercised in determining how to respond (or whether to respond at all to a <message/> request). For examples, the responding entity SHOULD NOT *automatically* (i.e. without first asking its human user) either respond to the initiating entity's request or return a specific error unless the initiating entity is subscribing to the responding entity's presence (and the responding entity's presence is not currently "invisible" to the initiating entity). Note: There should be no need for the responding entity's client to consult its block list, since if the initiating entity is on the list then the responding entity would not receive any requests from the initiating entity anyway.

4 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [9].

5 XMPP Registrar Considerations

In order for Jabber entities to adequately leverage **Data Forms** (e.g., by using machine-readable fields), it is RECOMMENDED to register standard x-data fields with the XMPP Registrar via the mechanisms defined in Field Standardization for Data Forms (XEP-0068) [10]. Whether to do so for any given features and options shall be determined by the "using protocol".

6 XML Schema

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='http://jabber.org/protocol/feature-neg'
    xmlns='http://jabber.org/protocol/feature-neg'
    elementFormDefault='qualified'>

  <xs:import namespace='jabber:x:data'
             schemaLocation='http://xmpp.org/schemas/x-data.xsd'/>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0020: http://www.xmpp.org/extensions/xep-0020.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='feature'>
    <xs:complexType>
      <xs:sequence xmlns:data='jabber:x:data'>
        <xs:element ref='data:x'/>
      </xs:sequence>
    </xs:complexType>
```

[9] 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/>.

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

```
  </xs:element>

</xs:schema>
```

7 Author Note

Peter Millard, the primary author of this specification from version 0.1 through version 1.4, died on April 26, 2006. The remaining authors are thankful for Peter's work on this specification.