XEP-0304: Whitespace Keepalive Negotiation

Ming Ji
mailto:mingj@cisco.com
xmpp:mingj@cisco.com

2011-08-18
Version 0.1

Status	Type	Short Name
Deferred	Standards Track	NOT_YET_ASSIGNED

This specification defines a method for negotiating how to send keepalives in XMPP.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

RFC 6120 [1] specifies that XMPP servers and clients can send whitespace characters between first-level elements of an XML stream as a way to maintain the state of the stream. These "whitespace keepalives" are widely used on the XMPP network. However, currently it is not possible to negotiate the frequency of sending keepalives, or even whether to send keepalives at all (the server simply sends them according to its own schedule). Because certain kinds of devices might not want to use keepalives, or might wish to receive keepalives more frequently or less frequently than the server's default, this specification defines a method for negotiating how to send whitespace keepalives between an XMPP server and an XMPP client (this method could also be used between two servers, although that usage is expected to be rare).

2 Protocol

The protocol defined in this document enables a client negotiate the time interval between whitespace keepalives, within a range determined by the server. Normally, the client starts the negotiation since not all kinds of the client support the feature.
The protocol defines a keepalive element and the normal negotaition process is quite simple, demonstrated as following:

1. Server announces keepalive feature with a range of time intervals.
2. Client starts the keepalive negotiation by specifying a time interval.
3. Server checks the value and relpies with success or failure.

Then server and client send whitespace keepalive to each other at the agreed time interval.

3 Use Cases

3.1 Stream Feature

During the stream negotiation process, the server can advertise this feature. The feature is negotiated after the resource binding. (See Recommended Order of Stream Feature Negotiation (XEP-0170) [2] regarding the recommended order of stream feature negotiation.)

Listing 1: Server lists feature in the stream negotiation stage

```
C: <stream:stream
```

[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>.

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

C: <stream:stream
            from='example.com'
            id='KskA2O2BEn5mL7mABTq9X3DTPHO44vAMaIg1nG9O1vo'
            version='1.0'
            xmlns='jabber:client'
            xmlns:stream='http://etherx.jabber.org/streams'>
S:  <stream:features>
        <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/>
        <keepalive xmlns='urn:xmpp:keepalive:0'>
          <interval min='60' max='300'/>
        </keepalive>
    </stream:features>
```

3.2 Negotiate Keepalive

Client negotiates the keepalive feature by providing an interval value based on server limits and its own condition. The interval SHOULD be a positive interger, and zero is invalid.

Listing 2: Client sends its negotiating keepalive time interval

```
<iq from='client@example.com/foo' id='p03ns61g' to='example.com' type=
    'set'>
  <keepalive xmlns='urn:xmpp:keepalive:0'>
    <interval>60</interval>
  </keepalive>
</iq>
```

The server checks the keepalive interval value and returns a success:

Listing 3: Server returns success of keepalive negotiation

```
<iq from='example.com' id='p03ns61g' to='client@example.com/foo' type=
    'result'/>
```

If a problem occurs, the server returns an error (in this example, the client sent a value outside the range of the server's preferences):

Listing 4: Server returns failure of keepalive negotiation

```
<iq from='example.com' id='p03ns61g' to='client@example.com/foo' type=
    'error'>
  <error type='cancel'>
```

```
    <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

Then both the server and the client send whitespace keepalives at the interval of the negotiated value. The server SHOULD check to see if has received keepalives and MAY drop the connection if it has not received a keepalive for a period of time significantly longer than the negotiated value (the client MAY also implement this behavior).

4 Implementation Notes

Several things need to be noted:

1. Both parties SHOULD keep a record of the keepalive status and honor the negotiated value when allowing stream resumption and session recreation. See Stream Management (XEP-0198) [3]
2. The length of the keepalive interval depends on the service type and network environment. Implementations are encouraged to use appropriate values based on implementation and deployment experience.

5 Security Considerations

During negotiation, the server MUST check the keepalive interval value and reject any invalid values.

6 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [4].

[3] XEP-0198: Stream Management <https://xmpp.org/extensions/xep-0198.html>.

[4] 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/>.

7 XMPP Registrar Considerations

7.1 Protocol Namespaces

The XMPP Registrar [5] is requested to issue an initial namespace 'urn:xmpp:keepalive:0'

7.2 Stream Features

The XMPP Registrar [6] is requested to issue an initial stream feature namespace 'urn:xmpp:keepalive:0'.

8 XML Schema

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

<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'
            targetNamespace='urn:xmpp:keepalive:0'
            xmlns='urn:xmpp:keepalive:0'
            elementFormDefault='unqualified'>

  <xsd:element name='keepalive'>
    <xsd:complexType>
      <xsd:attribute name='min' type='positiveShort' use='optional'/>
      <xsd:attribute name='max' type='positiveShort' use='optional'/>
      <xsd:assert test='@min le @max'/>
      <xsd:element name='interval' type='positiveShort' minOccurs='0'
          maxOccurs='1'/>
    </xsd:complexType>
  </xsd:element>

  <xsd:simpleType name='positiveShort'>
    <xsd:restriction base='xsd:unsignedShort'>
      <xsd:minExclusive value='0'/>
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>
```

[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/>.

[6] 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/>.

9 Acknowledgements

Thanks to Matt Miller and Peter Saint-Andre for their feedback.