XEP-0251: Jingle Session Transfer

Marian Podgoreanu
mailto:marian@null.ro

Paul Chitescu
mailto:paulc@null.ro

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

2009-10-05
Version 0.2

Status	Type	Short Name
Deferred	Standards Track	NOT_YET_ASSIGNED

This specification defines an extension to XMPP Jingle for transferring a session (such as a voice call) from one person to another.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2018 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).

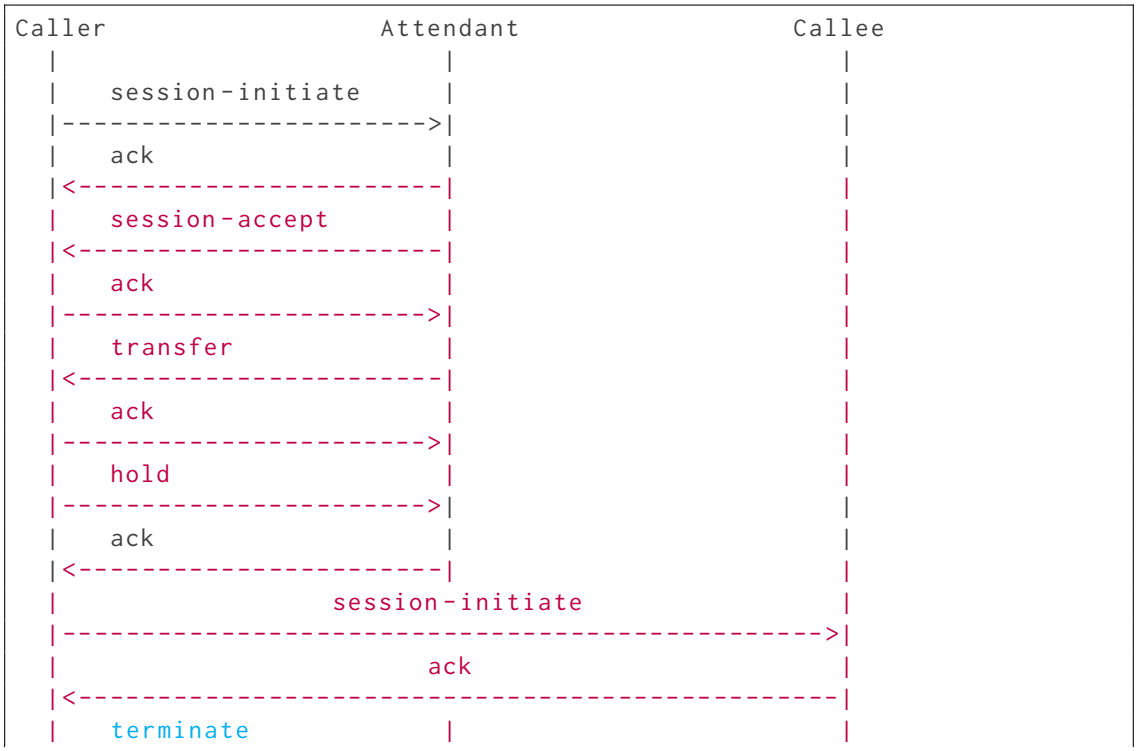
Contents

1 Introduction

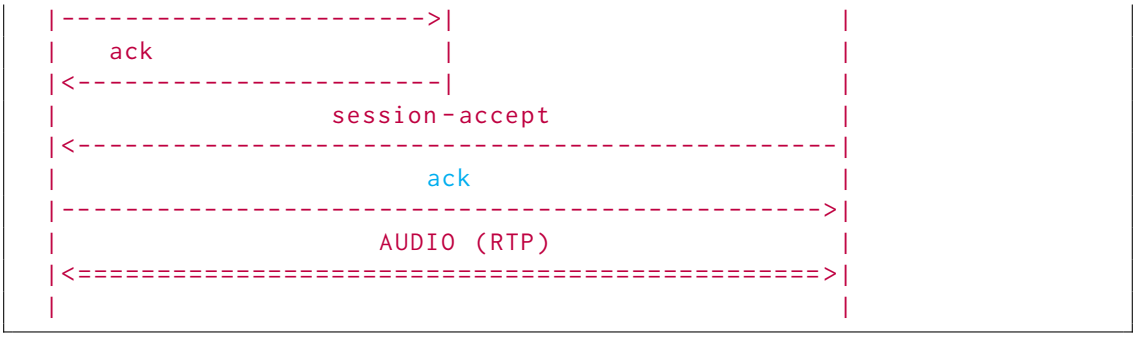
The Jingle (XEP-0166) [1] extensions to XMPP provide a technology for setup, management, and teardown of multimedia sessions between two entities, with an initial focus on voice over Internet Protocol (VoIP). By design, Jingle has been kept relatively simple and it does not cover the kind of advanced features that are available on the public switched telephone network (PSTN) and traditional private branch exchange (PBX) systems. However, because Jingle and XMPP itself provide an extensible technology for the real-time exchange of XML data, more advanced use cases can be defined through additional extensions. This document specifies one such extension, for the transfer of a session from one entity to another entity using either attended transfer or unattended transfer (for the difference between these scenarios, see for example RFC 5359 [2]). Although this extension will likely be used mainly in the context of VoIP interactions, it could also be used for any Jingle application type, such as video chat or screen sharing.

2 Unattended Transfer

The session flow for negotiating an unattended transfer is as follows:



[1] XEP-0166: Jingle <https://xmpp.org/extensions/xep-0166.html>.
[2] RFC 5359: Session Initiation Protocol Service Examples <http://tools.ietf.org/html/rfc5359>.



The protocol flow is shown below, where the caller is "caller@example.net", the attendant is "attendant@office.example.com", and the callee is "boss@execs.example.com".
First the caller initiates a normal call to the attendant.

Listing 1: Caller calls attendant

```
<iq from='caller@example.net/phone'
    id='i3hd81k8'
    to='attendant@office.example.com/desk'
    type='set'>
 <jingle xmlns='urn:xmpp:jingle:1'
         action='session-initiate'
         initiator='caller@example.net/phone'
         sid='851ba2'>
   <content creator='initiator'
            disposition='session'
            name='urgent-communication'>
     <description xmlns='urn:xmpp:jingle:apps:rtp:1'
                  media='audio'>
       <payload-type id='0' name='PCMU' />
     </description>
     <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'>
       ...
     </transport>
   </content>
 </jingle>
</iq>
```

The attendant's phone then acknowledges the session request.

Listing 2: Attendant acks session-initiate

```
<iq from='attendant@office.example.com/desk'
    id='i3hd81k8'
    to='caller@example.net/phone'
    type='result'/>
```

Next the attendant answers the call.

Listing 3: Attendant sends session-accept

```
<iq from='attendant@office.example.com/desk'
    id='k1d26dcv'
    to='caller@example.net/phone'
    type='set'>
 <jingle xmlns='urn:xmpp:jingle:1'
         action='session-accept'
         initiator='caller@example.net/phone'
         responder='attendant@office.example.com/desk'
         sid='851ba2'>
   <content creator='initiator'
            disposition='session'
            name='urgent-communication'>
     <description xmlns='urn:xmpp:jingle:apps:rtp:1'
                  media='audio'>
       <payload-type id='0' name='PCMU' />
     </description>
     <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'>
       ...
     </transport>
 </jingle>
</iq>
```

The caller acknowledges the session-accept.

Listing 4: Caller acks session-accept

```
<iq from='caller@example.net/phone'
    id='k1d26dcv'
    to='attendant@office.example.com/desk'
    type='result'/>
```

Now the attendant decides to transfer the call. It does this by sending a Jingle action of type session-info to the caller, specifying the address of the callee via a <transfer/> element qualified by the "urn:xmpp:jingle:transfer:0" namespace (see Namespace Versioning regarding the possibility of incrementing the version number).

Listing 5: Attendant transfers the call

```
<iq from='attendant@office.example.com/desk'
    id='a0pl3v76'
    to='caller@example.net/phone'
    type='set'>
 <jingle xmlns='urn:xmpp:jingle:1'
         action='session-info'
```

```
        initiator='caller@example.net/phone'
        responder='attendant@office.example.com/desk'
        sid='851ba2'>
   <transfer xmlns='urn:xmpp:jingle:transfer:0'
             to='boss@execs.example.com/phone'/>
 </jingle>
</iq>
```

If the caller understands the transfer request, it acknowledges the request (if not, it MUST return a <feature-not-implemented/> error as specified in XEP-0166).

Listing 6: Caller acks transfer request

```
<iq from='attendant@office.example.com/desk'
    to='caller@example.net/phone'
    id='a0pl3v76'
    type='result'/>
```

Now the caller puts the attendant on hold.

Listing 7: Caller puts attendant on hold

```
<iq from='attendant@office.example.com/desk'
    id='o4bd91v4'
    to='caller@example.net/phone'
    type='set'>
 <jingle xmlns='urn:xmpp:jingle:1'
         action='session-info'
         initiator='caller@example.net/phone'
         sid='851ba2'>
   <hold xmlns='urn:xmpp:jingle:apps:rtp:info:1'/>
 </jingle>
</iq>
```

While the attendant is on hold, the caller initiates a new call to the callee. The session-initiation request includes a <transfer/> element that specifies the attendant's address.

Listing 8: Caller initiates new call to callee

```
<iq from='caller@example.net/phone'
    id='r7y2nxv3'
    to='boss@execs.example.com/phone'
    type='set'>
 <jingle xmlns='urn:xmpp:jingle:1'
         action='session-initiate'
         initiator='caller@example.net/phone'
         sid='1a332d'>
   <content creator='initiator'
            disposition='session'
```

```
            name='urgent-communication'>
    <description xmlns='urn:xmpp:jingle:apps:rtp:1'
                 media='audio'>
      <payload-type id='0' name='PCMU' />
    </description>
    <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'>
      ...
    </transport>
  </content>
  <transfer xmlns='urn:xmpp:jingle:transfer:0'
            from='attendant@office.example.com/desk'/>
 </jingle>
</iq>
```

The callee acknowledges the call.

Listing 9: Callee acks session-initiate

```
<iq from='boss@execs.example.com/phone'
    to='caller@example.net/phone'
    id='r7y2nxv3'
    type='result'/>
```

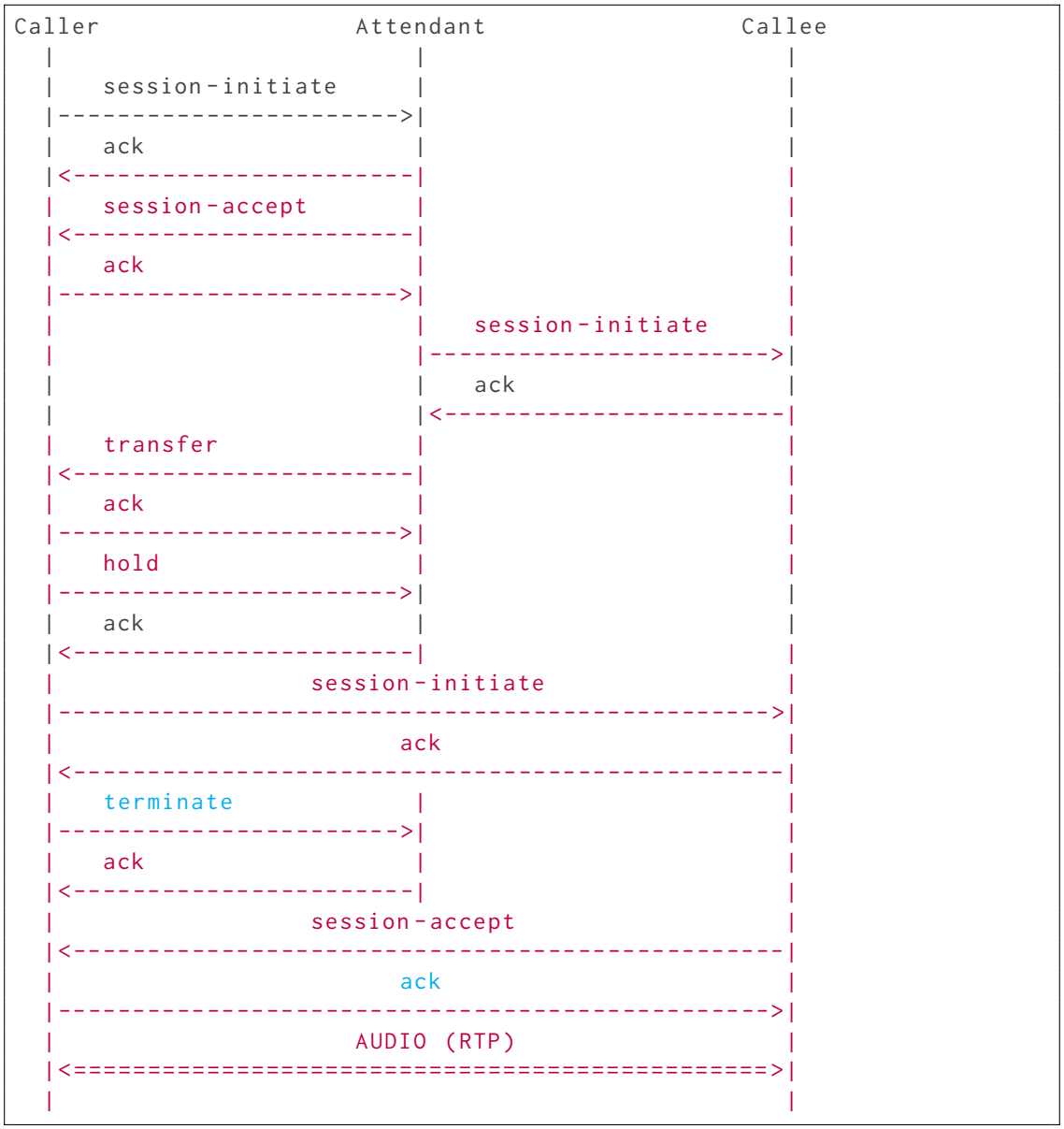
Now the caller's phone detects the successful transfer, so it hangs up on the attendant:

Listing 10: Caller terminates session with attendant

```
<iq from='caller@example.net/phone'
    id='hg34cx20'
    to='attendant@office.example.com/desk'
    type='set'>
 <jingle xmlns='urn:xmpp:jingle:1'
         action='session-terminate'
         initiator='attendant@office.example.com/desk'
         sid='851ba2'>
   <reason>
     <success/>
     <transferred xmlns='urn:xmpp:jingle:transfer:0'/>
     <text>Unattended transfer success</text>
   </reason>
 </jingle>
</iq>
```

3 Attended Transfer

The session flow for negotiating an attended transfer is as follows:



The protocol flow is shown below, where the caller is "caller@example.net", the attendant is "attendant@office.example.com", and the callee is "boss@execs.example.com".
First the caller initiates a normal call to the attendant.

Listing 11: Caller calls attendant

```
<iq from='caller@example.net/phone'
    id='m4hs861b'
    to='attendant@office.example.com/desk'
    type='set'>
```

```
 <jingle xmlns='urn:xmpp:jingle:1'
         action='session-initiate'
         initiator='caller@example.net/phone'
         sid='851ba2'>
   <content creator='initiator'
            disposition='session'
            name='urgent-communication'>
     <description xmlns='urn:xmpp:jingle:apps:rtp:1'
                  media='audio'>
       <payload-type id='0' name='PCMU' />
     </description>
     <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'>
       ...
     </transport>
   </content>
 </jingle>
</iq>
```

The attendant's phone then acknowledges the session request.

Listing 12: Attendant acks session-initiate

```
<iq from='attendant@office.example.com/desk'
    to='caller@example.net/phone'
    id='m4hs861b'
    type='result'/>
```

Next the attendant answers the call.

Listing 13: Attendant sends session-accept

```
<iq from='attendant@office.example.com/desk'
    to='caller@example.net/phone'
    id='u72bx793'
    type='set'>
 <jingle xmlns='urn:xmpp:jingle:1'
         action='session-accept'
         initiator='caller@example.net/phone'
         responder='attendant@office.example.com/desk'
         sid='851ba2'>
   <content creator='initiator'
            disposition='session'
            name='urgent-communication'>
     <description xmlns='urn:xmpp:jingle:apps:rtp:1'
                  media='audio'>
       <payload-type id='0' name='PCMU' />
     </description>
     <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'>
       ...
```

```
    </transport>
 </jingle>
</iq>
```

The caller acknowledges the session-accept.

Listing 14: Caller acks session-accept

```
<iq from='caller@example.net/phone'
    id='u72bx793'
    to='attendant@office.example.com/desk'
    type='result'/>
```

Next the attendant makes a call to the callee for the purpose of completing an attended transfer. Before doing so, the attendant SHOULD verify that the callee supports Jingle session transfer, as described under Determining Support.

Listing 15: Attendant calls callee

```
<iq from='attendant@office.example.com/desk'
    id='t57caw2r'
    to='boss@execs.example.com/phone'
    type='set'>
 <jingle xmlns='urn:xmpp:jingle:1'
         action='session-initiate'
         initiator='attendant@office.example.com/desk'
         sid='663e9f'>
   <content creator='initiator'
            disposition='session'
            name='urgent-communication'>
     <description xmlns='urn:xmpp:jingle:apps:rtp:1'
                  media='audio'>
       <payload-type id='0' name='PCMU' />
     </description>
     <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'>
       ...
     </transport>
   </content>
 </jingle>
</iq>
```

The callee acknowledges the session-initiate.

Listing 16: Callee acks session-initiate

```
<iq from='boss@execs.example.com/phone'
    id='t57caw2r'
    to='attendant@office.example.com/desk'
    type='result'/>
```

Now the attendant transfers the call by sending a session-info action to the caller containing details about the attendant's session with the callee.

Listing 17: Attendant transfers the call

```
<iq from='attendant@office.example.com/desk'
    id='p4hslk49'
    to='caller@example.net/phone'
    type='set'>
 <jingle xmlns='urn:xmpp:jingle:1'
         action='session-info'
         initiator='caller@example.net/phone'
         responder='attendant@office.example.com/desk'
         sid='851ba2'>
   <transfer xmlns='urn:xmpp:jingle:transfer:0'
             from='attendant@office.example.com/desk'
             sid='663e9f'
             to='boss@execs.example.com/phone'/>
 </jingle>
</iq>
```

If the caller understands the transfer request, it acknowledges the request (if not, it MUST return a <feature-not-implemented/> error as specified in XEP-0166).

Listing 18: Caller acks transfer request

```
<iq from='attendant@office.example.com/desk'
    id='p4hslk49'
    to='caller@example.net/phone'
    type='result'/>
```

Now the caller puts the attendant on hold.

Listing 19: Caller puts attendant on hold

```
<iq from='attendant@office.example.com/desk'
    id='o4bd91v4'
    to='caller@example.net/phone'
    type='set'>
 <jingle xmlns='urn:xmpp:jingle:1'
         action='session-info'
         initiator='caller@example.net/phone'
         sid='851ba2'>
   <hold xmlns='urn:xmpp:jingle:apps:rtp:info:1'/>
 </jingle>
</iq>
```

While the attendant is on hold, the caller initiates a new call to the callee. The session-initiation request includes a <transfer/> element that specifies the attendant's address and

the SessionID of the attendant's session with the callee.

Listing 20: Caller initiates new call to callee

```
<iq from='caller@example.net/phone'
    id='w93b461v'
    to='boss@execs.example.com/phone'
    type='set'>
 <jingle xmlns='urn:xmpp:jingle:1'
         action='session-initiate'
         initiator='caller@example.net/phone'
         sid='851ba2'>
   <content creator='initiator'
            disposition='session'
            name='urgent-communication'>
     <description xmlns='urn:xmpp:jingle:apps:rtp:1'
                  media='audio'>
       <payload-type id='0' name='PCMU' />
     </description>
     <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'>
       ...
     </transport>
   </content>
   <transfer xmlns='urn:xmpp:jingle:transfer:0'
             from='attendant@office.example.com/desk'
             sid='663e9f'
             to='boss@execs.example.com/phone'/>
 </jingle>
</iq>
```

The callee identifies an active session with the same from+to+sid and replaces that with the incoming call, so it hangs up on the existing session with the attendant.

Listing 21: Callee hangs up on attendant

```
<iq from='boss@execs.example.com/phone'
    id='yh2f36s5'
    to='attendant@office.example.com/desk'
    type='set'>
 <jingle xmlns='urn:xmpp:jingle:1'
         action='session-terminate'
         initiator='attendant@office.example.com/desk'
         sid='663e9f'>
   <reason>
     <success/>
     <transferred xmlns='urn:xmpp:jingle:transfer:0'/>
     <text>Attended transfer success</text>
   </reason>
 </jingle>
```

```
</iq>
```

The callee then acknowledges the session request from the caller.

Listing 22: Callee acks session-initiate

```
<iq from='boss@execs.example.com/phone'
    to='caller@example.net/phone'
    id='w93b461v'
    type='result'/>
```

Now the caller's phone detects the successful transfer, so it hangs up on the attendant:

Listing 23: Caller terminates session with attendant

```
<iq from='caller@example.net/phone'
    id='yh2f36s5'
    to='attendant@office.example.com/desk'
    type='set'>
 <jingle xmlns='urn:xmpp:jingle:1'
         action='session-terminate'
         initiator='attendant@office.example.com/desk'
         sid='851ba2'>
   <reason>
     <success/>
     <transferred xmlns='urn:xmpp:jingle:transfer:0'/>
     <text>Unattended transfer success</text>
   </reason>
 </jingle>
</iq>
```

4 Determining Support

If an entity supports session transfers, it MUST advertise that fact by returning a feature of "urn:xmpp:jingle:transfer:0" (see Namespace Versioning regarding the possibility of incrementing the version number) in response to Service Discovery (XEP-0030) [3] information requests.

Listing 24: Service discovery information request

```
<iq from='caller@example.net/phone
␣␣␣␣id='u891vad3'
␣␣␣␣to='attendant@office.example.com/desk'
␣␣␣␣type='get'>
␣␣<query␣xmlns='http://jabber.org/protocol/disco#info'/>
```

[3] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

```
</iq>
```

Listing 25: Service discovery information response

```
<iq from='attendant@office.example.com/desk'
    id='u891vad3'
    to='caller@example.net/phone
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='urn:xmpp:jingle:1'/>
    <feature var='urn:xmpp:jingle:transfer:0'/>
  </query>
</iq>
```

In order for an application to determine whether an entity supports this protocol, where possible it SHOULD use the dynamic, presence-based profile of service discovery defined in Entity Capabilities (XEP-0115) [4]. However, if an application has not received entity capabilities information from an entity, it SHOULD use explicit service discovery instead.

5 Security Considerations

In unattended transfer, the callee has no way to verify that the attendant specified in the session-initiate request received from the caller was actually involved in the transaction. This implies that:

1. A malicious caller could attribute its session-initiate request to an attendant, thus discrediting the attendant in the eyes of the callee.
2. A malicious attendant (or malicious code that has infected an attendant's legitimate client) could "transfer" all session requests it receives to the callee and disavow any responsibility.

6 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [5].

[4] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.

[5] 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

This specification defines the following XML namespace:

- urn:xmpp:jingle:transfer:0

Upon advancement of this specification from a status of Experimental to a status of Draft, the XMPP Registrar [6] shall add the foregoing namespaces to the registry located at <https://xmpp.org/registrar/namespaces.html>, as described in Section 4 of XMPP Registrar Function (XEP-0053) [7].

7.2 Protocol Versioning

If the protocol defined in this specification undergoes a revision that is not fully backwards-compatible with an older version, the XMPP Registrar shall increment the protocol version number found at the end of the XML namespaces defined herein, as described in Section 4 of XEP-0053.

8 XML Schemas

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:jingle:transfer:0'
    xmlns='urn:xmpp:jingle:transfer:0'
    elementFormDefault='qualified'>

  <xs:element name='transfer'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='empty'>
          <xs:attribute name='from' type='xs:string' use='optional'/>
          <xs:attribute name='sid' type='xs:string' use='optional'/>
          <xs:attribute name='to' type='xs:string' use='optional'/>
        </xs:extension>
      </xs:simpleContent>
```

[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/>.

[7] XEP-0053: XMPP Registrar Function <https://xmpp.org/extensions/xep-0053.html>.

```
    </xs:complexType>
  </xs:element>

  <xs:element name='transferred' type='empty'/>

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

</xs:schema>
```

9 Acknowledgements

Thanks to Robert McQueen for his feedback.