XEP-0130: Waiting Lists

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

Yehezkel Dallal
mailto:yehezkeld@followap.com

Alexandre Nolle
mailto:anolle@francetelecom.com

Jean-Louis Seguineau
mailto:jean-louis.seguineau@antepo.com
xmpp:jlseguineau@im.antepo.com

Mark Troyer
mailto:mtroyer@jabber.com
xmpp:mtroyer@corp.jabber.com

Valerie Mercier
mailto:valerie.mercier@orange-ftgroup.com
xmpp:vmercier@jabber.com

2012-04-18
Version 1.4

Status	Type	Short Name
Deprecated	Historical	waitinglist

This document defines an XMPP protocol extension that enables a user to add a non-IM user to a waiting list and be informed when the contact creates an IM account.

Contents

1 Introduction

An IM user may want to be informed when a contact creates an IM account. If the user knows some information about the contact (e.g., a phone number or email address), the user's service can use that information to place the contact on a "waiting list", then inform the user when the contact creates an IM account. This document defines an extension to XMPP Core [1] and XMPP IM [2] that enables such "waiting list" functionality, including the ability to add contacts on other domains if service providers agree to interoperate (e.g., to add a contact who uses a different mobile telephony service provider).
Note: The protocol defined herein is currently in use at several large service providers in Europe. Others are welcome to use the protocol.

2 Glossary

Contact A person with whom an IM User seeks to communicate, identified by a URI such as <tel:PhoneNumber> (see RFC 3966 RFC 3966: The tel URI for Telephone Numbers <http://tools.ietf.org/html/rfc3966>.) or <mailto:EmailAddress> (see RFC 2368 RFC 2368: The mailto URL scheme <http://tools.ietf.org/html/rfc2368>.).

Customer A person who is contracted for services with a ServiceProvider.

IM User Any Customer who has registered for instant messaging services.

InteropPartner Any company that agrees to interoperate using the protocol defined herein.

JID The unique identifier of an IM User in the XMPP protocol. Outside the context of an IM session, a JID is of the form (<localpart@domain.tld> or <domain.tld>) ("bare JID"); within the context of an IM session, a JID is of the form (<localpart@domain.tld/resource> or <domain.tld/resource>) ("full JID").

ServiceProvider A company that provides telephony or email services to a Customer.

URI A Uniform Resource Identifier as defined in RFC 3986 RFC 3986: Uniform Resource Identifiers (URI): Generic Syntax <http://tools.ietf.org/html/rfc3986>.. Specific URI schemes that may be useful in this specification include 'tel:', 'mailto:', and 'sip:', but any URI scheme may be used.

Waiting List A list of Contacts whom an entity (IM User or InteropPartner) is waiting to hear about regarding their status as instant messaging users.

WaitingListService An XMPP service that maintains Waiting lists for IM Users and/or InteropPartners.

[1] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[2] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.

3 Requirements

This protocol is designed so that an IM User can:

1. Request the user's current Waiting List
2. Add a Contact to a local WaitingList (based on some URI associated with the Contact)
3. Receive notification from a local WaitingListService if the Contact has (or subsequently creates) an IM account
4. Remove a Contact from the Waiting List

In addition, this protocol is designed so that a ServiceProvider can:

1. Request the service's current WaitingList
2. Add a Contact to a WaitingList at an InteroPartner (based on some URI associated with the Contact)
3. Receive notification from the InteropPartner if the Contact has (or subsequently creates) an IM account
4. Remove a Contact from the Waiting List

4 Use Cases

4.1 IM User Retrieves Current WaitingList

Before adding or removing Contacts from its WaitingList, an IM User SHOULD retrieve its current WaitingList. The activity flow is as follows:

4.1.1 Primary Flow

1. IM User discovers WaitingListService hosted by ServiceProvider [A1]; it is RECOMMENDED to do this immediately after logging in.
2. IM User requests current WaitingList from WaitingListService.
3. WaitingListService returns WaitingList to IM User, including any items for which JIDs have been discovered. [A2]

4.1.2 Alternate Flows

1. ServiceProvider does not host a WaitingListService:
    a) Use Case Ends unsuccessfully.
2. IM User does not have a Waiting List:
    a) WaitingListService returns <item-not-found/> error to IM User.
    b) Use Case Ends unsuccessfully.

4.2 IM User Adds Contact to WaitingList

An IM User may know a URI for a Contact (e.g., a phone number or email address) but not the Contact's JID. In order to subscribe to the Contact's presence or otherwise communicate with the Contact over an instant messaging system, the IM User first needs to discover the Contact's JID based on a URI for the Contact. However, the Contact may not yet have an IM account. Because the IM User may therefore need to wait until the Contact creates an account, the IM User needs to add the Contact to a WaitingList. The activity flow is as follows:

4.2.1 Primary Flow

1. IM User completes IM User Retrieves Current WaitingList use case.
2. IM User requests addition of Contact to WaitingList based on Contact's URI.
3. WaitingListService determines that the URI scheme is supported. [A1]
4. WaitingListService determines that the information provided is a valid URI for that URI scheme. [A2]
5. WaitingListService determines that Contact's URI does not belong to a person served by ServiceProvider. [A3]
6. WaitingListService acknowledges addition of Contact to IM User's WaitingList.
7. WaitingListService discovers WaitingListServices hosted by one or more InteropPartners.
8. WaitingListService queries one or more InteropPartner's WaitingListServices for JID associated with URI.
9. InteropPartner's WaitingListService determines that Contact's URI belongs to a person served by that partner. [A4]
10. InteropPartner's WaitingListService determines that Contact is an IM User. [A5]
11. InteropPartner's WaitingListService informs ServiceProvider's WaitingListService of JID associated with Contact's URI. [A6] [A10]

12. ServiceProvider's WaitingListService informs IM User of Contact's JID. [A8]
13. IM User completes IM User Removes Contact from WaitingList use case.
14. Use Case Ends.

4.2.2 Alternate Flows

1. The URI scheme is not supported:
   a) WaitingListService sends <bad-request/> error to IM User and does not add contact to WaitingList.
   b) Use Case Ends unsuccessfully.
2. The information provided is not a valid URI:
   a) WaitingListService sends <not-acceptable/> error to IM User and does not add contact to WaitingList.
   b) Use Case Ends unsuccessfully.
3. URI belongs to person served by ServiceProvider:
   a) WaitingListService determines that Contact is an IM User registered with ServiceProvider [A7].
   b) WaitingListService informs IM User of Contact's JID. [A9]
   c) IM User completes IM User Removes Contact from WaitingList use case.
   d) Use Case Ends.
4. URI does not belong to a person served by InteropPartner:
   a) InteropPartner sends <item-not-found/> error to WaitingListService.
   b) If all InteropPartners queried return <item-not-found/> error, WaitingListService sends <item-not-found/> error (or local equivalent) to IM User.
   c) IM User completes IM User Removes Contact from WaitingList use case.
   d) Use Case Ends unsuccessfully.
5. Contact is not an IM User registered with InteropPartner:
   a) InteropPartner records and acknowledges WaitingListService's request for JID associated with URI.
   b) OPTIONALLY, InteropPartner invites Contact to register as an IM User.
   c) Contact registers.
   d) InteropPartner informs Service Provider's WaitingListService of JID associated with Contact's URI.
   e) ServiceProvider's WaitingListService informs all IM Users who requested JID associated with Contact's URI.

   f) IM User completes IM User Removes Contact from WaitingList use case.
   g) Use Case Ends.

6. InteropPartner refuses to provide service to ServiceProvider:
   a) InteropPartner's WaitingListService sends <not-authorized/> error to ServiceProvider's WaitingListService.
   b) If all other InteropPartners also return errors, WaitingListService returns <item-not-found/> error to IM User.
   c) Use Case Ends unsuccessfully.

7. Contact is not an IM User registered with ServiceProvider:
   a) WaitingListService records IM User's request for JID associated with URI.
   b) OPTIONALLY, WaitingListService invites Contact to register as an IM User.
   c) Contact registers.
   d) WaitingListService informs all IM Users who requested JID associated with Contact's URI.
   e) IM User completes IM User Removes Contact from WaitingList use case.
   f) Use Case Ends.

8. Contact's URI is not handled by any ServiceProvider:
   a) WaitingListService informs all IM Users who requested JID associated with Contact's URI that no InteropPartner services Contact's URI.
   b) IM User completes IM User Removes Contact from WaitingList use case.
   c) Use Case Ends unsuccessfully.

9. IM User completes IM User Removes Contact from WaitingList use case.
   a) ServiceProvider's WaitingListService removes item from WaitingList.
   b) Use Case Ends unsuccessfully.

10. All Users Remove Contact from Their WaitingLists
   a) ServiceProvider's WaitingListService removes item from WaitingList at InteropPartner's WaitingListService.
   b) Use Case Ends unsuccessfully.

4.3 IM User Removes Contact from WaitingList

An IM User should remove a contact from the WaitingList after the IM User Adds Contact to WaitingList use case ends (either successfully or unsuccessfully), and may remove a contact from the WaitingList at any other time.

4.3.1 Primary Flow

1. IM User sends removal request to WaitingListService.
2. WaitingListService removes IM User's request for JID associated with URI.
3. WaitingListService informs IM User of successful removal [A1].
4. WaitingListService sends removal request to appropriate InteropPartner's WaitingList-Service [A2].
5. InteropPartner's WaitingListService determines that URI belongs to a person served by that partner.
6. InteropPartner's WaitingListService removes ServiceProvider's WaitingListService's request for JID.
7. InteropPartner's WaitingListService informs ServiceProvider's WaitingListService of successful removal.
8. Use Case Ends.

4.3.2 Alternate Flows

1. IM User never requested JID associated with URI:
   a) WaitingListService sends <item-not-found/> error to IM User.
   b) Use Case Ends.
2. Contact URI is served by WaitingListService or IM User was not the only person who requested the JID:
   a) Use Case Ends.

5 Protocol

5.1 IM User Interaction With WaitingListService

This section of the document is provided for the sake of domains that implement XMPP as their local protocol; domains that implement another protocol will use their service-specific protocol to complete the user-to-domain interaction.

5.1.1 IM User Retrieves Current WaitingList

It is RECOMMENDED for an IM User's client to retrieve the WaitingList immediately after logging in. However, first it must discover its local WaitingListService. An IM User MAY use either Service Discovery (XEP-0030) [3] or the deprecated Agent Information (XEP-0094) [4]

[3]XEP-0030: Service Discovery <http://xmpp.org/extensions/xep-0030.html>.
[4]XEP-0094: Agent Information <http://xmpp.org/extensions/xep-0094.html>.

protocol.

Listing 1: IM User Discovers WaitingListService by Sending Agent Information Request to its Server

```
<iq type='get' id='agent1'>
  <query xmlns='jabber:iq:agents'/>
</iq>
```

Listing 2: Server Returns Address of its WaitingListService

```
<iq type='result' id='agent1'>
  <query xmlns='jabber:iq:agents'>
    ...
    <agent jid='waitlist.service-provider.com'>
      <name>Waiting List Service</name>
      <service>waitinglist</service>
    </agent>
    ...
  </query>
</iq>
```

Listing 3: IM User Discovers WaitingListService by Sending Service Discovery Request to its Server

```
<iq type='get' id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#items'/>
</iq>
```

Listing 4: Server Returns Address of its WaitingListService

```
<iq type='result' id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#items'>
    ...
    <item jid='waitlist.service-provider.com'
          name='Waiting␣List␣Service'/>
    ...
  </query>
</iq>
```

Listing 5: IM User Queries WaitingListService for Detailed Information

```
<iq type='get'
    from='user@service-provider.com/resource'
    to='waitlist.service-provider.com'
    id='disco2'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

The WaitingListService SHOULD return detailed information about the service it provides, including the URI schemes it supports (see also the Service Discovery Features section of this document).

Listing 6: WaitingListService Returns Detailed Information

```
<iq type='result'
    from='waitlist.service-provider.com'
    to='user@service-provider.com/resource'
    id='disco2'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='directory' type='waitinglist'/>
    <feature var='http://jabber.org/protocol/waitinglist'/>
    <feature var='http://jabber.org/protocol/waitinglist/schemes/
       mailto'/>
    <feature var='http://jabber.org/protocol/waitinglist/schemes/tel'/
       >
  </query>
</iq>
```

Once an IM User has discovered the WaitingListService, the user's client SHOULD request its current Waiting List. This is done by sending an IQ-get to the WaitingListService containing an empty <query/> element qualified by the 'http://jabber.org/protocol/waitinglist' namespace:

Listing 7: IM User Requests its Current WaitingList

```
<iq type='get'
    from='user@service-provider.com/resource'
    to='waitlist.service-provider.com'
    id='request1'>
  <query xmlns='http://jabber.org/protocol/waitinglist'/>
</iq>
```

Upon request, the WaitingListService MUST return the current WaitingList to the IM User:

Listing 8: WaitingListService Returns WaitingList to IM User

```
<iq type='result'
    from='waitlist.service-provider.com'
    to='user@service-provider.com/resource'
    id='request1'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item id='12345'>
      <uri scheme='tel'>3033083282</uri>
      <name>PSA</name>
    </item>
    <item id='23456'>
      <uri scheme='mailto'>editor@xmpp.org</uri>
      <name>XMPP Extensions Editor</name>
    </item>
  </query>
</iq>
```

Each ItemID MUST be unique within the scope of the client's WaitingList items. The value of the ItemID is an opaque string; an implementation MAY assign semantic meaning to the ItemID (e.g., id="John Smith (mobile)" rather than id="12345"), but such meaning is implementation-specific and outside the scope of the protocol defined herein. The user MAY include a <name/> element containing a natural-language name for the Contact.
The WaitingList MAY contain an item for which a JID has been discovered.

Listing 9: IM User Asks for its WaitingList including Newly Discovered JID

```
<iq type='get'
    from='user@service-provider.com/resource'
    to='waitlist.service-provider.com'
    id='jidask1'>
  <query xmlns='http://jabber.org/protocol/waitinglist'/>
</iq>

<iq type='result'
    from='waitlist.service-provider.com'
    to='user@service-provider.com/resource'
    id='jidask1'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item id='12345' jid='stpeter@jabber.org'>
      <uri scheme='tel'>3033083282</uri>
      <name>PSA</name>
    </item>
    <item id='23456'>
      <uri scheme='mailto'>editor@xmpp.org</uri>
      <name>XMPP Extensions Editor</name>
    </item>
  </query>
</iq>
```

5.1.2 IM User Adds Contact to WaitingList

Once an IM User's client has discovered the WaitingListService and requested the user's WaitingList, the user can add Contacts to the WaitingList based on the Contact's URI. (Note: This document uses the example of phone numbers via the 'tel' URI scheme, but the same rules apply to WaitingList items based on email addresses or other URI schemes.)

Listing 10: IM User Requests Addition of Contact to WaitingList

```
<iq type='set'
```

```
    to='waitlist.service-provider.com'
    id='waitinglist1'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item>
      <uri scheme='tel'>contact-number</uri>
      <name>contact-name</name>
    </item>
  </query>
</iq>
```

As described below, various error conditions may occur. (For information about error syntax, refer to RFC 6120 and Error Condition Mappings (XEP-0086) [5].)
If the IM User provided a URI whose scheme is not supported, WaitingListService MUST return a <bad-request/> error to the IM User and MUST NOT add the Contact to the WaitingList.

Listing 11: WaitingListService Returns <bad-request/> Error to IM User

```
<iq type='error'
    from='waitlist.service-provider.com'
    to='user@service-provider.com/resource'
    id='waitinglist1'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item>
      <uri scheme='tag'>shakespeare.lit,2005-08:waitlist1</uri>
      <name>contact-name</name>
    </item>
  </query>
  <error code='400' type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the IM User included a JID in the request, WaitingListService MUST return a <bad-request/> error to IM User and MUST NOT add the Contact to the WaitingList. (Note: A WaitingList-Service MUST NOT return a non-XMPP URI to an IM User based on the Contact's JID; see the Security Considerations section of this document.)

Listing 12: WaitingListService Returns <bad-request/> Error to IM User

```
<iq type='error'
    from='waitlist.service-provider.com'
    to='user@service-provider.com/resource'
    id='waitinglist1'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item jid='some-jid'>
      <uri scheme='tel'>contact-number</uri>
      <name>contact-name</name>
```

[5] XEP-0086: Error Condition Mappings <http://xmpp.org/extensions/xep-0086.html>.

```
    </item>
  </query>
  <error code='400' type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the IM User provided an invalid URI (e.g., a phone number with too many digits or an email address with no '@' character), WaitingListService MUST return a <not-acceptable/> error to the IM User and MUST NOT add the Contact to the WaitingList.

Listing 13: WaitingListService Returns <not-acceptable/> Error to IM User

```
<iq type='error'
    from='waitlist.service-provider.com'
    to='user@service-provider.com/resource'
    id='waitinglist1'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item>
      <uri scheme='tel'>+1234563033083283</uri>
      <name>contact-name</name>
    </item>
  </query>
  <error code='406' type='modify'>
    <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If one of the foregoing errors was generated (all of which have a type of "modify"), IM User SHOULD modify the request and re-submit it.
If none of the "modify" errors was generated, WaitingListService MUST inform the IM User that the request was successfully received, including a unique ID number for the new WaitingList item.

Listing 14: WaitingListService Informs IM User that Request was Received

```
<iq type='result'
    from='waitlist.service-provider.com'
    to='user@service-provider.com/resource'
    id='waitinglist1'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item id='34567'/>
  </query>
</iq>
```

If none of the "modify" errors was generated and WaitingListService knows Contact JID when the IQ result is returned to the user (e.g., because Contact is served by ServiceProvider), WaitingListService MAY include the WaitingList item in the IQ result: [6]

Listing 15: WaitingListService Returns IQ Result to IM User (With Contact JID)

```xml
<iq type='result'
    from='waitlist.service-provider.com'
    to='user@service-provider.com/resource'
    id='waitinglist1'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item id='34567' jid='contact@service-provider.com'>
      <uri scheme='tel'>contact-number</uri>
      <name>contact-name</name>
    </item>
  </query>
</iq>
```

If none of the "modify" errors was generated and WaitingListService does not know Contact JID when the IQ result is returned to the user, it needs to contact InteropPartners in order to determine if the Contact is associated with one of the InteropPartners. Thus before it returns the Contact JID to the IM User, it needs to wait for the one of the InteropPartners to return Contact JID or for all of the InteropPartners to return errors.
If all of the InteropPartners return an error of type "cancel" (typically <item-not-found/> and/or <not-authorized/>) to WaitingListService, WaitingListService MUST return an <item-not-found/> error (or local equivalent) to the IM User (and IM User SHOULD complete IM User Removes Contact from WaitingList use case).

Listing 16: WaitingListService Returns <item-not-found/> Error to IM User

```xml
<message
    type='error'
    from='waitlist.service-provider.com'
    to='user@service-provider.com/resource'
    id='waitinglist1'>
  <waitlist xmlns='http://jabber.org/protocol/waitinglist'>
    <item>
      <uri scheme='tel'>+1234563033083283</uri>
      <name>contact-name</name>
    </item>
  </waitlist>
  <error code='404' type='cancel'>
    <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</message>
```

If the connection to at least one of the InteropPartners times out (a <remote-server-timeout/> error), WaitingListService MUST return an IQ-result as described above (indicating that the

[6]Even if WaitingListService returns Contact JID in the IQ-result, it MUST also send a "JID push" message.

request was received) and resend the request to the InteropPartners that timed out. If connections continue to time out (over some configurable time period and for some configurable number of retries), WaitingListService SHOULD then return a <remote-server-timeout/> error to IM User via a ”JID push” message as shown below.
If InterPartner’s WaitingListService knows the Contact JID, it sends it to ServiceProvider’s WaitingListService as shown in the ServiceProvider’s WaitingListService Adds Contact to WaitingList section of this document.
If WaitingListService knows Contact JID (or learns Contact JID from InteropPartner), it MUST inform IM User through a ”JID push” message, which consists of a message stanza that contains a <waitlist/> element qualified by the ’http://jabber.org/protocol/waitinglist’ namespace: [7]

Listing 17: WaitingListService Pushes Contact’s JID to IM User

```
<message
    from='waitlist.service-provider.com'
    to='user@service-provider.com'>
  <body>This message contains a WaitingList item.</body>
  <waitlist xmlns='http://jabber.org/protocol/waitinglist'>
    <item id='34567' jid='contact@service-provider.com'>
      <uri scheme='tel'>contact-number</uri>
      <name>contact-name</name>
    </item>
  </waitlist>
</message>
```

Note: The JID push uses an XMPP <message/> stanza because the WaitingListService has no knowledge of the user’s presence and therefore cannot assume that an <iq/> stanza will be received by the user at a specific resource.
If WaitingListService learns that Contact’s URI is not handled by any InteropPartner, it MUST inform IM User through a ”JID push” message:

Listing 18: WaitingListService Informs IM User that No InteropPartner Handles Contact’s URI

```
<message
    from='waitlist.service-provider.com'
    to='user@service-provider.com'>
  <body>Sorry, we cannot find this contact.</body>
  <waitlist xmlns='http://jabber.org/protocol/waitinglist'>
    <item id='34567'
          jid='contact@service-provider.com'
```

[7]When waiting list information is included in a message stanza, the root element for the ’http://jabber.org/protocol/waitinglist’ namespace is <waitlist/> rather than <query/> (as used within IQ stanzas). This disparity is historical and tracks the protocol syntax that was most widely implemented, as defined in version 0.4 of this specification. In the interest of interoperability, the IQ usage was changed back to <query/> in version 1.1 of this specification. If this document were not historical, the root element usage would be harmonized to use only the <waitlist/> element.

```
          type='error'>
      <uri scheme='tel'>contact-number</uri>
      <name>contact-name</name>
      <error code='404'
             type='cancel'
             xmlns='jabber:client'>
        <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
      </error>
    </item>
  </waitlist>
</message>
```

After receiving the "JID push" message, IM User SHOULD complete the IM User Removes Contact from WaitingList use case.

5.1.3 IM User Removes Contact from WaitingList

In order to remove the item from the WaitingList, the IM User MUST complete the Remove Contact from WaitingList use case.

Listing 19: IM User Sends Removal Request to WaitingListService

```
<iq type='set'
    from='user@service-provider.com/resource'
    to='waitlist.service-provider.com'
    id='remove1'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item id='34567'>
      <remove/>
    </item>
  </query>
</iq>
```

If WaitingListService previously recorded request, WaitingListService removes request from list and returns result to IM User.

Listing 20: WaitingListService Returns Result to IM User

```
<iq type='result'
    from='waitlist.service-provider.com'
    to='user@service-provider.com/resource'
    id='remove1'/>
```

If WaitingListService did not previously record this request, WaitingListService MUST return an <item-not-found/> error to the IM User.

Listing 21: WaitingListService Returns <item-not-found/> Error to IM User

```
<iq type='error' id='remove1'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item id='34567'>
      <remove/>
    </item>
  </query>
  <error code='404' type='cancel'>
    <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

5.2 WaitingListService Interaction With InteropPartners

This section of the document describes the inter-domain protocol for communication between WaitingListServices. The protocol defined in this section MUST be implemented by ServiceProviders.
A ServiceProvider's WaitingListService MUST be configured with a "whitelist" of InteropPartner's WaitingListServices with which it communicates. Therefore service discovery SHOULD NOT be necessary. However, if necessary it MAY use either the Agent Information protocol or the Service Discovery protocol as described in the following examples.
Note: The InteropPartner's WaitingListService is not required to be hosted by InteropPartner, and could be hosted by a third party (e.g., a neutral phone number translation service). In this case, InteropPartner would simply advertise 'waitlist.third-party.com' as its WaitingListService.

5.2.1 ServiceProvider's WaitingListService Retrieves Current WaitingList

Listing 22: ServiceProvider Discovers InteropPartner's WaitingListService by Sending Agent Information Request to InteropPartner

```
<iq type='get'
    from='waitlist.service-provider.com'
    to='interop-partner.com'
    id='agent2'>
  <query xmlns='jabber:iq:agents'/>
</iq>
```

Listing 23: InteropPartner Returns Address of its WaitingListService

```
<iq type='result'
    from='interop-partner.com'
    to='waitlist.service-provider.com'
    id='agent2'>
  <query xmlns='jabber:iq:agents'>
    ...
    <agent jid='waitlist.interop-partner.com'>
```

```
      <name>Waiting List Service</name>
      <service>waitinglist</service>
    </agent>
    ...
  </query>
</iq>
```

Listing 24: ServiceProvider Discovers InteropPartner's WaitingListService by Sending Service Discovery Request to InteropPartner

```
<iq type='get'
    from='waitlist.service-provider.com'
    to='interop-partner.com'
    id='disco3'>
  <query xmlns='http://jabber.org/protocol/disco#items'/>
</iq>
```

Listing 25: InteropPartner Returns Address of its WaitingListService

```
<iq type='result' id='disco3'>
  <query xmlns='http://jabber.org/protocol/disco#items'>
    ...
    <item jid='waitlist.service-provider.com'
          name='Waiting␣List␣Service'/>
    ...
  </query>
</iq>
```

Listing 26: Service Provider Queries InteropPartner's WaitingListService for Detailed Information

```
<iq type='get'
    from='waitlist.service-provider.com'
    to='waitlist.interop-partner.com'
    id='disco4'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 27: InteropPartner's WaitingListService Returns Detailed Information

```
<iq type='result'
    from='waitlist.interop-partner.com'
    to='waitlist.service-provider.com'
    id='disco4'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='directory'
              type='waitinglist'/>
    <feature var='http://jabber.org/protocol/waitinglist'/>
  </query>
</iq>
```

5.2.2 ServiceProvider’s WaitingListService Adds Contact to WaitingList

Once a ServiceProvider’s WaitingListService has discovered the InteropPartner’s WaitingList-Service and requested its WaitingList, the ServiceProvider’s WaitingListService can add items to its WaitingList based on URI.

Listing 28: ServiceProvider’s WaitingListService Adds New Item to WaitingList

```
<iq type='set'
    from='waitlist.service-provider.com'
    to='waitlist.interop-partner.com'
    id='waitinglist2'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item>
      <uri scheme='tel'>contact-number</uri>
    </item>
  </query>
</iq>
```

If InteropPartner refuses to provide service to ServiceProvider, it MUST return a <not-authorized/> error to the ServiceProvider:

Listing 29: InteropPartner Returns <not-authorized/> Error to ServiceProvider

```
<iq type='error'
    from='waitlist.interop-partner.com'
    to='waitlist.service-provider.com'
    id='waitinglist2'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item>
      <uri scheme='tel'>contact-number</uri>
    </item>
  </query>
  <error code='401' type='cancel'>
    <not-authorized xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If Contact’s URI is not associated with a person served by this InteropPartner, the Interop-Partner MUST return an <item-not-found/> error to the ServiceProvider.

Listing 30: InteropPartner Returns <item-not-found/> Error to ServiceProvider

```
<iq type='error'
    from='waitlist.interop-partner.com'
    to='waitlist.service-provider.com'
    id='waitinglist2'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item>
```

```
      <uri scheme='tel'>contact-number</uri>
    </item>
  </query>
  <error code='404' type='cancel'>
    <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If ServiceProvider's WaitingListService receives <not-authorized/> and/or <item-not-found/> errors from all InteropPartners, it returns a <item-not-found/> error to IM User:

Listing 31: WaitingListService Returns <item-not-found/> Error to IM User

```
<message
    type='error'
    from='waitlist.service-provider.com'
    to='user@service-provider.com/resource'
    id='waitinglist1'>
  <waitlist xmlns='http://jabber.org/protocol/waitinglist'>
    <item>
      <uri scheme='tel'>+1234563033083283</uri>
      <name>contact-name</name>
    </item>
  </waitlist>
  <error code='404' type='cancel'>
    <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</message>
```

If Contact's URI is associated with a person served by this InteropPartner, InteropPartner MUST return acknowledgement of the WaitingList addition to the ServiceProvider's WaitingListService.

Listing 32: InteropPartner's WaitingListService Acknowledges Receipt

```
<iq type='result'
    from='waitlist.interop-partner.com'
    to='waitlist.service-provider.com'
    id='waitinglist2'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item id='34567'/>
  </query>
</iq>
```

If Contact is an IM User served by InteropPartner, InteropPartner's WaitingListService pushes Contact's JID to ServiceProvider's WaitingListService.

Listing 33: InteropPartner’s WaitingListService Pushes Contact’s JID to ServiceProvider’s WaitingListService

```
<iq type='set'
    from='waitlist.interop-partner.com'
    to='waitlist.service-provider.com'
    id='jidpush1'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item id='34567' jid='user@domain'>
      <uri scheme='tel'>contact-number</uri>
    </item>
  </query>
</iq>
```

Listing 34: ServiceProvider’s WaitingListService Acknowledges Receipt of JID Push

```
<iq type='result'
    from='waitlist.service-provider.com'
    to='waitlist.interop-partner.com'
    id='jidpush1'/>
```

After receiving acknowledgement (but not before), InteropPartner’s WaitingListService MUST remove that item from the WaitingList for the ServiceProvider’s WaitingListService.

5.2.3 ServiceProvider’s WaitingListService Removes Contact from WaitingList

Listing 35: ServiceProvider Requests Removal of Item from WaitingList

```
<iq type='set'
    from='waitlist.service-provider.com'
    to='waitlist.interop-partner.com'
    id='remove2'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item id='34567'>
      <remove/>
    </item>
  </query>
</iq>
```

If item exists on WaitingList, InteropPartner’s WaitingListService removes item from list and returns result to ServiceProvider’s WaitingListService.

Listing 36: InteropPartner Returns Result to ServiceProvider

```
<iq type='result'
    from='waitlist.interop-partner.com'
    to='waitlist.service-provider.com'
    id='remove2'/>
```

If item does not exist on WaitingList, InteropPartner's WaitingListService MUST return an <item-not-found/> error to the ServiceProvider's WaitingListService.

Listing 37: InteropPartner Returns <item-not-found/> Error to ServiceProvider

```
<iq type='error'
    from='waitlist.interop-partner.com'
    to='waitlist.service-provider.com'
    id='remove2'>
  <query xmlns='http://jabber.org/protocol/waitinglist'>
    <item id='34567'>
      <remove/>
    </item>
  </query>
  <error code='404' type='cancel'>
    <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

6 Implementation Notes

1. Protocols and mechanisms for inviting a Contact to register as an IM User are out of scope for this document and shall be determined by each InteropPartner individually.

2. A ServiceProvider's WaitingListService MUST record which of its IM Users have requested the JID associated with Contact's URI, and an InteropPartner's WaitingListService MUST record that Service Provider's WaitingListService (not User) has requested JID associated with Contact's URI. Therefore when Contact registers, InteropPartner's WaitingListService informs its local users as well as ServiceProvider's WaitingListService, and ServiceProvider's WaitingListService informs its local users.

3. The InteropPartner's WaitingListService is not required to be hosted by InteropPartner, and could be hosted by a third party (e.g., a neutral phone number translation service). In this case, InteropPartner would simply advertise 'waitlist.third-party.com' as its WaitingListService.

4. Once an IM User learns a Contact's JID, the IM User MAY send a normal subscription request to the Contact, setting the "to" address to Contact's JID. This interaction is defined in the base XMPP specifications and is out of scope for this document.

5. For historical reasons, implementations MUST support the older Agent Information protocol (XEP-0094) and SHOULD support Service Discovery (XEP-0030). Note well

that the Agent Information protocol will eventually be deprecated in favor of Service Discovery.

6. An IM User's client receives WaitingList information either through a "JID push" message (received from WaitingListService at any time) or in the IQ result received after requesting the WaitingList (since one or more of the WaitingList items may contain a JID). (The same rule applies to a ServiceProvider's WaitingListService that receives an IQ set from an InteropPartner's WaitingListService.)

7. When an IM User logs in, the user's client SHOULD request the current WaitingList.

8. Although the examples in this document show the hostname of the WaitingListService as 'waitlist.third-party.com' (etc.), this is for convenience only; the hostname MAY be any valid DNS hostname.

9. When sending JID pushes, an implementation MAY specify a message type of 'headline', which in some deployments will prevent such messages from being stored offline for later delivery.

10. It can happen that WaitingListService does not receive a reply from InteropPartner within a certain amount of time or the connection to InteropPartner times out. Because such behavior is often transient, WaitingListService MAY attempt to reconnect and then resend the request (although any retry logic to handle these cases is a matter of implementation). However, WaitingListService SHOULD NOT return an <item-not-found/> error to IM User unless it knows definitively that the Contact's InteropPartner is permanently unavailable, since returning an <item-not-found/> error in response to temporary connection timeouts is likely to be misleading.

7 Security Considerations

A ServiceProvider's WaitingListService MUST be configured with a "whitelist" of InteropPartners with which it communicates. The WaitingListService SHOULD NOT communicate with any InteropPartners that are not on the whitelist.
Requesting JIDs via WaitingLists is not bidirectional; i.e., a service MUST NOT allow an IM User to discover a Contact's non-XMPP URI based on the Contact's JID.
A service MAY require a Contact to approve the disclosure of the Contact's JID, either as a global preference or for each request; however, this is a local policy matter.

8 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [8].

9 XMPP Registrar Considerations

9.1 Protocol Namespaces

The XMPP Registrar [9] includes 'http://jabber.org/protocol/waitinglist' in its registry of protocol namespaces.

9.2 Service Discovery Identities

The Jabber Registar includes a type of "waitinglist" in the "directory" category in its registry of service discovery identities.

9.3 Service Discovery Features

The XMPP Registrar includes supported URI schemes in its registry of service discovery features. These features shall be of the form 'http://jabber.org/protocol/waitlist/schemes/SCHEME-NAME'.
This document registers the following two namespace names for URI schemes, but others MAY be registered in the future using standard registration procedures:

- http://jabber.org/protocol/waitlist/schemes/mailto
- http://jabber.org/protocol/waitlist/schemes/tel

10 XML Schema

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
```

[8] 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/>.

[9] 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/>.

```
    targetNamespace='http://jabber.org/protocol/waitinglist'
    xmlns='http://jabber.org/protocol/waitinglist'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0130: http://www.xmpp.org/extensions/xep-0130.html
    </xs:documentation>
  </xs:annotation>

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

  <xs:annotation>
    <xs:documentation>
      Note: there are two allowable root elements for the
      'http://jabber.org/protocol/waitinglist' namespace,
      query and waitlist. The query element is used within
      IQ stanzas and the waitlist element is used within
      message stanzas. See XEP-0130 for details.
    </xs:documentation>
  </xs:annotation>

  <xs:element name='waitlist'>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='item'
                    minOccurs='0'
                    maxOccurs='unbounded'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='query'>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='item'
                    minOccurs='0'
                    maxOccurs='unbounded'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='item'>
    <xs:complexType>
      <xs:choice minOccurs='0'
                 maxOccurs='unbounded'
```

```
                  xmlns:xmpp='jabber:client'>
      <xs:sequence>
        <xs:element ref='uri'/>
        <xs:element ref='name' minOccurs='0'/>
        <xs:element ref='xmpp:error' minOccurs='0'/>
      </xs:sequence>
      <xs:element ref='remove'/>
    </xs:choice>
    <xs:attribute name='id'
                  type='xs:string'
                  use='optional'/>
    <xs:attribute name='jid'
                  type='xs:string'
                  use='optional'/>
    <xs:attribute name='type'
                  use='optional'>
      <xs:simpleType>
        <xs:restriction base='xs:NCName'>
          <xs:enumeration value='error'/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

<xs:element name='uri'>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base='xs:string'>
        <xs:attribute name='scheme'
                      type='xs:NCName'
                      use='required'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name='name' type='string1023'/>

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

<xs:simpleType name='string1023'>
  <xs:restriction base='xs:string'>
    <xs:maxLength value='1023'/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name='empty'>
  <xs:restriction base='xs:string'>
```

```
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```