XEP-0407: Mediated Information eXchange (MIX): Miscellaneous Capabilities

Kevin Smith
mailto:kevin.smith@isode.com
xmpp:kevin.smith@isode.com

Steve Kille
mailto:steve.kille@isode.com
xmpp:steve.kille@isode.com

2020-11-03
Version 0.1.2

Status	Type	Short Name
Deferred	Standards Track	MIX-MISC

This document defines an extension to Mediated Information eXchange (MIX) specified in XEP-0369. It specifies a number of independent optional capabilities that MAY be used with MIX.

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 <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 Mediated Information eXchange (MIX) protocol framework and core capabilities are specified in Mediated Information eXchange (MIX) (XEP-0369) [1] (MIX-CORE). This specification defines six extensions to this core:

1. Avatar Publishing.
2. Nick Registration.
3. Retracting a Message.
4. Telling another user about a channel.
5. Invitation by reference.
6. Converting 1:1 conversation to a channel.

These extensions are independent. An implementation of this specification should identify clearly which extensions are implemented.

2 Avatar Publishing

Mediated Information eXchange (MIX) (XEP-0369) [2] defines a framework of nodes associated with each channel, where new nodes can be added to provide new services. Two nodes defined in this specification MAY bey used to support Avatars. These nodes and their use is defined in User Avatar (XEP-0084) [3]. These nodes MAY be created as part of a MIX channel, where it is desired to publish an avatar associated with the channel.

Name	Node	Description
Avatar Data	'urn:xmpp:avatar:data'	For publishing an Avatar
Avatar Metadata	'urn:xmpp:avatar:metadata'	For publishing Avatar metadata

3 Registering a Nick

A nick MAY be associated with a user's bare JID. A user can register a nick with the MIX service. Nick registration can be used ensure that a user is able to use the same nick in all

[1] XEP-0369: Mediated Information eXchange (MIX) <https://xmpp.org/extensions/xep-0369.html>.
[2] XEP-0369: Mediated Information eXchange (MIX) <https://xmpp.org/extensions/xep-0369.html>.
[3] XEP-0084: User Avatar <https://xmpp.org/extensions/xep-0084.html>.

channels in the service and to prevent other users from using a registered nick. This can help achieve a consistent experience across a set of channels and prevent user confusion. Support for nick registration by a MIX service is OPTIONAL. Where nick registration is supported, nick registration MAY be OPTIONAL or MANDATORY. Where a user has registered a Nick with the MIX service, it MAY be used by each channel according to policy for the channel. A channel MAY enforce use of a registered nick. A channel MUST NOT use a registered nick for any other participant.
In order to determine if a Nick is allowed to be registered, the user MAY use disco to determine capabilities of the MIX service.

Listing 1: User Determines features of the MIX service

```
<iq type='get'
    from='hag66@shakespeare.example/UUID-a1j/7533'
    to='mix.shakespeare.example'
    id='7nve413p'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>

<iq type='result'
    to='hag66@shakespeare.example/UUID-a1j/7533'
    from='mix.shakespeare.example'
    id='7nve413p'>
    <query xmlns='http://jabber.org/protocol/disco#info'/>
        <feature var='urn:xmpp:mix:misc:0#nick-register'/>
    </query>
</iq>
```

The response will be a list of features of the MIX channel. If Nick Registration is supported, then the result set will include <feature var="urn:xmpp:mix:misc:0#nick-register"/>.
To register a nick with the MIX service the user sends a register command to the service. This is encoded as a <register/> child element of an <iq/> element. The <register/> element is qualified by the urn:xmpp:mix:misc:0' namespace. The nick is encoded in a <nick/> child element of the <register/> element.

Listing 2: User Registers with Service

```
<iq type='set'
    from='hag66@shakespeare.example/UUID-a1j/7533'
    to='mix.shakespeare.example'
    id='7nve413p'>
  <register xmlns='urn:xmpp:mix:misc:0'>
    <nick>thirdwitch</nick>
  </register>
</iq>
```

On success, the service informs the user of its nick. MIX SHOULD apply the "nickname" profile of the PRECIS OpaqueString class, as defined in RFC 7700 [4] to the requested nick. This means that nick that is issued might be different from the nick that was requested.
When an Nick is assigned, the MIX server MUST update the Participants Node in the channel to reflect this change. Any users subscribed to this node will be notified of the change of Nick. The following example shows an example of reporting successful Nick assignment.

Listing 3: Service Returns User of Nick

```
<iq type='result'
    to='mix.shakespeare.example'
    from='hag66@shakespeare.example/UUID-a1j/7533'
    id='7nve413p'>
  <register xmlns='urn:xmpp:mix:misc:0'>
    <nick>thirdwitch</nick>
  </register>
</iq>
```

If the requested nick is already taken and the MIX service does not assign an alternate nick, the MIX service MUST return a <conflict/> error:

Listing 4: Nick is Taken

```
<iq type='error'
    to='mix.shakespeare.example'
    from='hag66@shakespeare.example/UUID-a1j/7533'
    id='7nve413p'>
  <error type='cancel'>
    <conflict xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the register request does not contain a <nick/> element, then the MIX service MUST assign one. It is RECOMMENDED that the assigned nick is a UUID following RFC 4122 [5].

4 Retracting a Message

A MIX channel MAY support message retraction, where the sender of a messages or an authorized administrator deletes a message. A MIX channel MAY limit the time frame in which a message is allowed to be retracted, for example to prevent retraction of very old messages. When a messages is retracted the original message MAY be replaced by a tombstone. Message retraction is done by sending a special message that identifies the original message. This

[4] RFC 7700: Preparation, Enforcement, and Comparison of Internationalized Strings Representing Nicknames<http://tools.ietf.org/html/rfc7700>.
[5] RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace <http://tools.ietf.org/html/rfc4122>.

mechanism allows the retraction to be distributed on the same path as the original message so that all participating servers and clients MAY honour the retraction. The protocol to request retraction does this by adding to a message a <retract> element qualified by the 'urn:xmpp:mix:misc:0' namespace. A retract messages MUST NOT have a body. The <retract> element MUST include an 'id' attribute that holds the MAM-ID of the original message. The message sender will need to look up the MAM-ID. The MAM-ID is the convenient message identification for message recipients. A message and its retraction shown in the following example.

Listing 5: User Retracts a Message

```
<message from='hag66@shakespeare.example/UUID-a1j/7533'
         to='coven@mix.shakespeare.example'
         id='abcde'>
  <body> A Message I did not mean to send </body>
</message>

<message from='hag66@shakespeare.example/UUID-a1j/7533'
         to='coven@mix.shakespeare.example'
         id='92vax143g'>
  <retract id='77E07BB0-55CF-4BD4-890E-3F7C0E686BBD' xmlns='
      urn:xmpp:mix:misc:0'/>
</message>
```

The MIX channel will allow a user to retract a message sent by the user if the 'Allow User Message Retraction' option is configured. The MIX channel will allow an administrative user to retract any message if the user is in the group specified by the 'Administrator Message Retraction Rights' option.
If the retraction message is accepted, it MUST be distributed to channel participants. This will allow retraction to happen in the MAM archive of each channel participant and to reflect the retraction in client GUI. A client receiving a retraction message SHOULD ensure that the retracted message is no longer displayed to the end user.
Two approaches to message retraction can be used. In the first approach, the retracted message is simply removed. This is appropriate where retraction is provided as a user service and the user has rights to remove messages sent from the record.
The second approach is to leave a tombstone, which if taken MUST be done in the following manner. It is recommended to use a tombstone, as simply deleting the message may cause confusion with MAM queries. Use of a tombstone is appropriate where it is desired to leave a record of the message that was redacted. With this approach, the original message <body> is removed and replaced with a tombstone using the <retracted> element qualified by the 'urn:xmpp:mix:misc:0' namespace that shows the JID of user performing the retraction and the time of the retraction.

Listing 6: Retracted message tombstone in a MAM result

```
<message id='aeb213' to='juliet@capulet.example/UUID-e3r/9264'>
  <result xmlns='urn:xmpp:mam:2' queryid='f27' id='28482-98726-73623'>
```

```
    <forwarded xmlns='urn:xmpp:forward:0'>
      <delay xmlns='urn:xmpp:delay' stamp='2010-07-10T23:08:25Z'/>
      <message xmlns='jabber:client' from="hag66@shakespeare.example"
               to="macbeth@shakespeare.example">
        <retracted xmlns='urn:xmpp:mix:misc:0' by='hag66@shakespeare.
            example'
               time='2010-07-10T23:08:25Z'/>
      </message>
    </forwarded>
  </result>
</message>
```

5 Telling another User about a Channel

A convenient way to reference another channel is to use References (XEP-0372) [6] which enables the JID of a channel to be shared. This might be used simply to inform the message recipient about the channel or as mechanism to invite the user to join the channel. This is useful as an invitation mechanism to a channel that any user can join or where the invitee knows that the user is allowed to join (e.g., because the channel is for all users in an organization).

6 Inviting another User to join a Channel that the user does not have Permission to Join

Invitation by reference, as described in the previous section, is a convenient approach to invite a user to join a channel that the user has permission to join. This section describes the approach used when the inviter has permission to grant rights for the invitee to become a channel participant. This might be because the inviter is an administrator of the channel or the channel or if the inviter is a channel participant and the channel allows invitation by participants (this channel capability is controlled by the channel configuration variable 'Participation Addition by Invitation from Participant'). Invitation by reference is used to avoid cluttering the allowed node with JIDs of users who are invited to join, but do not accept the invitation. When a channel participant (the inviter) invites another user (the invitee) to join a channel, the following sequence of steps is followed:

1. The inviter checks using capability discovery that the invitee supports MIX.
2. The channel inviter sends to the channel requesting an invite for the invitee.
3. The channel sends an invitation to the inviter.

[6] XEP-0372: References <https://xmpp.org/extensions/xep-0372.html>.

4. The inviter sends the invitation to the invitee.
5. The invitee MAY use the invitation to join the channel.
6. The invitee MAY send a response to the inviter, indicating if the invitation was accepted or declined.

The first step is for the inviter to request an invitation from the channel. The invitation contains inviter, invitee and a token. The channel will evaluate if the inviter has rights to issue the invitation. This will be because the inviter is a channel administrator or if the inviter is a channel participant and the channel allows invitation by participants. If the inviter has rights to make the invitation, the channel will return a token. The token is a string that the channel can subsequently use to validate an invitation. The format of the token is not specified in this standard. The encoded token MAY reflect a validity time. The invitation request is encoded as an <invite/> child element of an <iq/> element. The <invite/> element is qualified by the 'urn:xmpp:mix:misc:0' namespace. <invite/> contains an <invitation/> child element, which contains <inviter/>, <invitee/>, <channel/> and <token/> child elements.

Listing 7: Inviter Requests and Receives Invitation

```
<iq from='hag66@shakespeare.example/UUID-h5z/0253'
    id='kl2fax27'
    to='coven@mix.shakespeare.example'
    type='get'>
  <invite xmlns='urn:xmpp:mix:misc:0'>
      <invitee>cat@shakespeare.example</invitee>
  </invite>
</iq>


<iq from='coven@mix.shakespeare.example'
    id='kl2fax27'
    to='hag66@shakespeare.example/UUID-h5z/0253'
    type='result'>
  <invite xmlns='urn:xmpp:mix:misc:0'>
     <invitation>
        <inviter>hag66@shakespeare.example</inviter>
        <invitee>cat@shakespeare.example</invitee>
        <channel>coven@mix.shakespeare.example</channel>
        <token>ABCDEF</token>
      </invitation>
  <invite/>
</iq>
```

The inviter can now send the invitee a message containing the invitation within the <message/> element, as shown in the following example.

Listing 8: Inviter sends Invitation to Invitee

```
<message from='hag66@shakespeare.example/UUID-h5z/0253'
    id='f5pp2toz'
    to='cat@shakespeare.example'>
    <body>Would you like to join the coven?</body>
    <invitation xmlns='urn:xmpp:mix:misc:0'>
        <inviter>hag66@shakespeare.example</inviter>
        <invitee>cat@shakespeare.example</invitee>
        <channel>coven@mix.shakespeare.example</channel>
        <token>ABCDEF</token>
   </invitation>
</message>
```

The invitation can now be used by the invitee to join a channel. The <invitation/> child element is simply added to the standard channel <join/> element, so that the channel can validate the invitation using the token. If the allowed node is present and the invitee is not matched against any item, the channel MUST add the invitee to the allowed node as part of the join.

Listing 9: User Joins a Channel with an Invitation

```
<iq type='set'
    from='cat@shakespeare.example'
    to='coven@mix.shakespeare.example'
    id='E6E10350-76CF-40C6-B91B-1EA08C332FC7'>
  <join xmlns='urn:xmpp:mix:misc:0'>
    <subscribe node='urn:xmpp:mix:nodes:messages'/>
    <invitation>
        <inviter>hag66@shakespeare.example</inviter>
        <invitee>cat@shakespeare.example</invitee>
        <channel>coven@mix.shakespeare.example</channel>
        <token>ABCDEF</token>
   </invitation>
  </join>
</iq>
```

The invitee MAY send an acknowledgement back to the inviter, noting the status of the invitation. This is encoded as an <invitation-ack/> child element of <message/> element. The <invitation-ack/> element is qualified by the 'urn:xmpp:mix:misc:0' namespace. The <invitation-ack/> has an <invitation/> child element that encodes the invitation being acknowledged and a <value/> child element to encode the acknowledgement value. <value/> has the following values:

- 'Joined': The invitee has joined the channel.
- 'Declined': The invitee is not taking up the invitation.

- 'Acknowledged': The invitation is acknowledged, without information on action taken or planned.

Listing 10: Invitee sends Acknowledgement to Inviter

```
<message from='cat@shakespeare.example/UUID-l1w/8813'
    id='b6p9llze'
    to='hag66@shakespeare.example/UUID-h5z/0253'>
    <body>No Thanks - too busy chasing mice....<body>
    <invitation-ack xmlns='urn:xmpp:mix:misc:0'>
        <value>Declined</value>
        <invitation>
           <inviter>hag66@shakespeare.example</inviter>
           <invitee>cat@shakespeare.example</invitee>
           <channel>coven@mix.shakespeare.example</channel>
           <token>ABCDEF</token>
         </invitation>
     </invitation-ack>
</iq>
```

7 Converting a 1:1 Conversation to a Channel

A common use case for an ad hoc channel is where two users are engaged in a 1:1 chat and wish to broaden the discussion. Prior to bringing more users into a channel, using standard invitation process, there is a need to create and populate a channel. The first step is for one of the two users to create an ad hoc channel, as described in the previous section. The other user will then be invited, and can switch to the new channel.
It can also be useful to share some or all of the messages from the 1:1 discussion into the new channel. The mechanism to do this is to forward messages to be shared to the channel using Stanza Forwarding (XEP-0297) [7]. A body SHOULD NOT be used in the outer message. Sharing history is optional. If history is shared, it MUST be done by the user creating the channel before the other user is invited. Any other use of forwarded messages MUST be treated as a channel participant forwarding a message to the channel and MUST NOT be treated as history sharing.

Listing 11: Forwarding a message to create History

```
<message from='hag66@shakespeare.example/UUID-a1j/7533'
         to='A1B2C345@mix.shakespeare.example'
         id='92vax143g'
         type='groupchat'>
         <forwarded xmlns='urn:xmpp:forward:0'>
          <delay xmlns='urn:xmpp:delay' stamp='2010-07-10T23:08:25Z'/>
          <message from='hag67@shakespeare.example/pda'
```

[7] XEP-0297: Stanza Forwarding <https://xmpp.org/extensions/xep-0297.html>.

```
                    id='0202197'
                    to='hag66@shakespeare.example/UUID-a1j/7533'
                    type='chat'
                    xmlns='jabber:client'>
                <body>Harpier cries: 'tis time, 'tis time.</body>
            </message>
        </forwarded>
</message>
```

There are a number of security considerations with sharing 1:1 history in a channel. There may be sensitive information in the 1:1 history, and the user sharing this history should ensure that none of this is sensitive, prior to sharing in this way. The user creating the channel has potential to inject history messages which were not part of the history. It is recommended that the second user joining the channel to validate that the messages match the history and to take appropriate action if they do not.

8 Internationalization Considerations

See considerations in Mediated Information eXchange (MIX) (XEP-0369) [8].

9 Security Considerations

See considerations in Mediated Information eXchange (MIX) (XEP-0369) [9].
When converting a 1:1 conversation to a channel there is potential to expose sensitive information and to present invalid information.

10 IANA Considerations

None.

11 XMPP Registrar Considerations

The urn:xmpp:mix namespace needs to be registered.

[8] XEP-0369: Mediated Information eXchange (MIX) <https://xmpp.org/extensions/xep-0369.html>.
[9] XEP-0369: Mediated Information eXchange (MIX) <https://xmpp.org/extensions/xep-0369.html>.

12 XML Schema

To be supplied when MIX progresses to proposed standard.

13 Acknowledgements

See Mediated Information eXchange (MIX) (XEP-0369) [10] for a list of contributors to the MIX Family of specifications.

[10] XEP-0369: Mediated Information eXchange (MIX) <https://xmpp.org/extensions/xep-0369.html>.