XEP-0329: File Information Sharing

Jefry Lagrange
mailto:jefry.reyes@gmail.com
xmpp:j.lagrange@jabber.org

Lance Stout
mailto:lance@andyet.com
xmpp:lance@lance.im

2017-09-11
Version 0.4

Status	Type	Short Name
Deferred	Standards Track	fis

This document specifies a simple extension to existing protocols that allows an entity to request information about files.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

XMPP extensions provide ways of transferring files between peers (such as Jingle File Transfer (XEP-0234) [1] and SI File Transfer (XEP-0096) [2]). However, file transfer is currently limited to needing that the transfer be initiated by the hosting entity. The Jingle File Transfer (XEP-0234) [3] extension, provides for a way to request files, but it requires the requesting entity to have information about the file being requested, so that it can be uniquely identified.
This document defines an extension which allows the request of information of files being offered by a hosting entity so they can later be requested in a file transfer; If the requesting entity is interested in the file.
File Sharing (XEP-0135) [4] is a previous extention that attempted to solve the same problem as this one, but unfortunetly it didn't generate much interest and it got deprecated. This extention aims be a modern replacement for XEP-0135.
IRC users have been able to bypass the limitations of the protocol by using bots that provide information of files and initiate transfer on command. A major downside of this method is that not every user is capable of sharing its files. The aim of this document is to provide a similar functionality while making it easier for users to offer and request information about files.
Microsoft's MSN proprietary IM client, used to provide similar functionality using "Sharing Folders", but this was replaced by Windows Live SkyDrive

2 Requirements

1. Enable a requesting entity to traverse the shared directory of an offering entity (REQUIRED)

2. Enable a requesting entity to get detailed information about files. (REQUIRED)

3 Assumptions

This protocol assumes the existence of one or more shared directories (either virtual or physical). The hosting entity must not advertise empty directories. The hosting entity is responsible of maintaining the structure of those directories (such as not allowing two files with the same name and preventing cycles within directories). The hosting entity is in no way required to present the same shared directories to different requesters. In fact, the reason multiple share directories are allowed, is to make it easier for implementers to manage permissions of who can see what files.

[1] XEP-0234: Jingle File Transfer <https://xmpp.org/extensions/xep-0234.html>.
[2] XEP-0096: SI File Transfer <https://xmpp.org/extensions/xep-0096.html>.
[3] XEP-0234: Jingle File Transfer <https://xmpp.org/extensions/xep-0234.html>.
[4] XEP-0135: File Sharing <https://xmpp.org/extensions/xep-0135.html>.

4 Getting Information About Files

4.1 Traversing Files

If a requesting entity wishes to know what files are being shared by an offering entity, it can do so by sending the following query:

Listing 1: Requester queries the root of the shared folder

```
<iq type='get'
    from='juliet@capulet.com/chamber'
    to='romeo@montague.net/home'
    id='1234'>
  <query xmlns="urn:xmpp:fis:0" />
</iq>
```

If the offering entity wishes to share files with the requesting entity, it may respond with a list of shared folders. It MUST not include any files in this response.

Listing 2: The offering entity responds with shared directories

```
<iq type='result'
    from='romeo@montague.net/home'
    to='juliet@capulet.com/chamber'
    id='1234'>
  <query xmlns="urn:xmpp:fis:0">
    <directory name='documents'/>
    <directory name='pics'/>
    <directory name='audio'/>
  </query>
</iq>
```

if the offering entity has no files to offer

Listing 3: The offering entity responds with no files

```
<iq type='result'
    from='romeo@montague.net/home'
    to='juliet@capulet.com/chamber'
    id='1234'>
  <query xmlns="urn:xmpp:fis:0" />
</iq>
```

Requesting the list of files and directories within a directory.

Listing 4: The requesting entity wants to know about a particular directory

```
<iq type='get'
```

```
    from='juliet@capulet.com/chamber'
    to='romeo@montague.net/home'
    id='1235'>
  <query xmlns="urn:xmpp:fis:0" node="documents" />
</iq>
```

When replying with a list of files, the offering entity can choose to either reply with verbose information on the file using the file attributes defined by Jingle File Transfer (XEP-0234) [5] or it may reply only with the 'name' attribute, which is required and MUST be included in every response.
It is RECOMENDED, when the list files to be sent is small, that a verbose response be made (in order to avoid going back and forth requesting information), and that a non-verbose reponse be made otherwise. This recomendation is made to save bandwidth.

Listing 5: The offering entity replies with information about a particular directory

```
<iq type='result'
    from='romeo@montague.net/home'
    to='juliet@capulet.com/chamber'
    id='1235'>
  <query xmlns="urn:xmpp:fis:0" node="documents">
    <file xmlns='urn:xmpp:jingle:apps:file-transfer:4'>
      <name>test.txt</name>
      <date>1969-07-21T02:56:15Z</date>
      <desc>This is a test. If this were a real file...</desc>
      <range/>
      <size>1022</size>
      <hash xmlns='urn:xmpp:hashes:1' algo='sha-1'>552
          da749930852c69ae5d2141d3766b1</hash>
    </file>
    <file xmlns='urn:xmpp:jingle:apps:file-transfer:4'>
      <name>test2.txt</name>
    </file>
    <directory name="secret␣docs" />
  </query>
</iq>
```

If the requesting entity wants to get detailed information about a file. It can do so by providing its full path.

Listing 6: The requesting entity wants to know about a particular file

```
<iq type='get'
    from='juliet@capulet.com/chamber'
    to='romeo@montague.net/home'
    id='1236'>
```

[5] XEP-0234: Jingle File Transfer <https://xmpp.org/extensions/xep-0234.html>.

```
  <query xmlns="urn:xmpp:fis:0" node="documents/test2.txt" />
</iq>
```

Listing 7: The offering entity responds with more information

```
<iq type='result'
    from='romeo@montague.net/home'
    to='juliet@capulet.com/chamber'
    id='1236'>
  <query xmlns="urn:xmpp:fis:0" node="test2.txt">
    <file xmlns='urn:xmpp:jingle:apps:file-transfer:4'>
      <name>test2.txt</name>
      <size>1000</size>
    </file>
  </query>
</iq>
```

5 Bandwidth Considerations

If a considerable amount of files are being shared by the offering entity, it may be the case that the offering entity response might be too be for the server to handle; As there might be a limitation on the size of the stanzas in the current stream. In order to solve this, extensions have been devised and their implementation are hereby recommended along with the implementation of this extension.

5.1 Using Result Set Management

Result Set Management (XEP-0059) [6] defines a way of limiting the results of a request. There are some considerations to use result sets along with this extension.
First, it is defined that the requesting entity is the one that sets the limit of the number of items that can be replied. So it is up to the requesting entity to choose a sensible number.
Second, since this protocol defines a way of handling the directory tree structure by allowing file tags to be children of a directory tags, it becomes difficult to define items for Result Set Management. Therefore, when responding to a request, the offering entity MUST NOT send directory tags with files as their children.

5.2 Out of Band Transfer

One obvious way to overcome the limitations of sending large stanzas in-band, is to transfer that information out of band. Out-of-Band Stream Data (XEP-0265) [7] could be used for that purpose. It is hereby RECOMMENDED its implementation when the offering entity has a

[6] XEP-0059: Result Set Management <https://xmpp.org/extensions/xep-0059.html>.
[7] XEP-0265: Out-of-Band Stream Data <https://xmpp.org/extensions/xep-0265.html>.

massive amount of files that would not be practical to advertise in-band.
It is further recommended that when using XEP-0265, the entire directory structure, along with all the files in the shared folder and subfolders, be exchanged in one single reply. Also, all the files attributes should be included. This is to avoid wasting bandwidth initiating out of band streams going back and forth.

6 Implementation Notes

6.1 File identification

As it was previously discussed, when requesting detailed information about a file, only the "name" attribute is required, but it is strongly RECOMMENDED that the hash attribute be included, in order to reduce the chances of sending the wrong file. When requesting the file to be transferred using Jingle File Transfer (XEP-0234) [8], the information that must be provided has to identify the file uniquely. It is then RECOMMENDED that when requesting a file, the full path of the file in the shared folder be included in the "name" attribute.

```
<iq type='get'
    from='juliet@capulet.com/chamber'
    to='romeo@montague.net/home'
    id='1237'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='juliet@capulet.com/chamber'
          sid='uj3b2'>
    <content creator='initiator' name='a-file-request' senders='
        responder'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:4'>
        <file>
          <name>pics/test4.png</name>
          <size>10740</size>
        </file>
      </description>
    </content>
  </jingle>
</iq>
```

6.2 File Sharing in MUCs

For the most part, discovering files in a MUC is exactly the same as what has been described in this document. However, it is RECOMMENDED that a participant in a MUC should have a single shared folder associated with the entire room, as opposed to advertise different files

[8] XEP-0234: Jingle File Transfer <https://xmpp.org/extensions/xep-0234.html>.

to different participants of the room. This is to reduce the complexity of the client software. Also, due to volatile nature of the participants in a room, keeping track of permissions is more trouble than what it is worth.

7 Security Considerations

A denial of service is possible by repeatedly requesting files. Implementers are advised to take this into consideration and include queues and limits into their implementations.