Stringflow Content Model (SFCM)

(Build Version: 1.0.0)

[Internal]

Introduction

XMPP as a protocol is designed to transport small XML packets across the network and it does a fantastic job at that; however it's not very suitable to transport large data packets particularly binary data such as images, audio, video and files.

As the media became integral part of modern conversations, it is almost unavoidable to have a complete conversation without transferring media files. XMPP came up with many extensions to support file transfer both in-band and out-of-band. As in-band file transfer, files are transported on the XMPP stream as Base64 decoded; while in case of out-of-band file transfer, XMPP relies on other technologies such as HTTP, SOCKS5 etc. which are pluggable into XMPP.

In-band file transfer uses XMPP byte stream to transport files which adversely impacts the real-time conversation experience (sometimes it takes long to transfer a file and the text messages following the file get delayed). On the other hand, out-of-band file transfer requires another service to be set up for file transfer which XMPP server will use to transfer files. This is a big over-head in terms of infrastructure and also in terms of technology. Additionally, XMPP extensions have very lengthy negotiations making the file transfer complex and error prone.

Stringflow as server technology offers a proprietary extension over XMPP (Stringflow Content Model) for file transfer. Stringflow Content Model (SFCM) makes use of MIME as transport type to transfer files. It is very similar to HTTP as HTTP also uses MIME as payload transport technology. The negotiation for file transfer takes place on primary Stream i.e. authenticated XMPP Stream and files are transferred over a MIME connection. Stringflow as a server technology manages both the connection types.

Additionally, Stringflow server acts as a mediator for file transfer as opposed to a relay (suggested in XMPP specifications). In order to make file transfer robust and truly asynchronous, files are first uploaded to server and later receiver pulls them from server; meaning for a file exchange to take place, it is not mandatory for both receiver and sender to be online at the same time. An argument can be made that the approach will result in increased bandwidth consumption; while the argument is valid, it is a small price for better conversational experience. And in case it's a deal-breaker for the application, Stringflow server can be configured to transfer files only when both sender and receiver are online avoiding file dumping on server (please refer to Stringflow Developer Guide for more information).

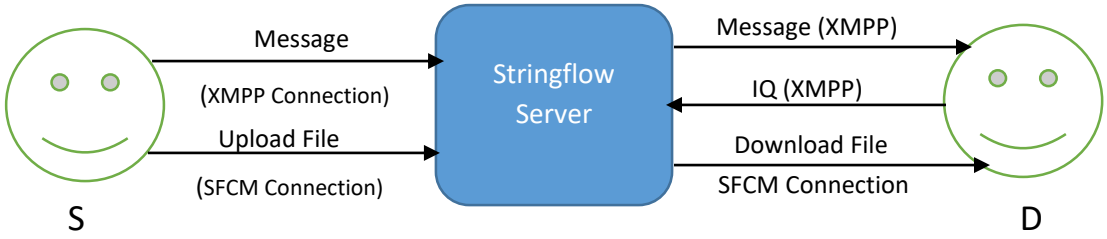
This document provides internal working of both Synchronous and Asynchronous File Transfer mechanisms supported by Stringflow server. Please note, asynchronous file transfer mode the default and recommended file transfer mechanism in Stringflow.

Asynchronous File Transfer

In case of asynchronous file transfer under SFCM, the file exchange between a sender and receiver has two separate and almost independent actions:

- Sender sending a file
- Receiver receiving the file

Let's assume user Shalaka wants to send a file to another user Dharmu; and both of them are on Stringflow infrastructure. Below diagram explains the data packet flow.



Sender Sending a File: In order to send a file to user Dharmu, Shalaka's SDK will execute two operations-

- Sending a Message packet with media details and thumb to user Dharmu
- Uploading the file to Stringflow server

The first operation (sending a Message stanza with media details) is executed over existing XMPP stream. The message packet sent is the extension of the Message stanza in XMPP specification and is processed as any other Message stanza. Below is an example of such Message packet.

```
<message>
      <media id='ftsksd-78snd-bsnd88-nsd9'>
            <content-type>image/jpeg</content-type>
            <thumb>Base64 thumbnail</thumb>
      </media>
</message>
```

The above Message packet is delivered to user Dharmu and the application will immediately display the received file thumb on Dharmu's device. At this point, the content of the file has not reached Dharmu's device.

Right after the Message has been sent, Shalaka's SDK will execute the second operation (uploading the file to server). The SDK will negotiate a SFCM connection with Stringflow server and the file content will be uploaded as a MIME message on the negotiated SFCM connection.

Receiver Receiving the File: As noted above, as soon as Shalaka sends a file to Dharmu, a Message stanza with file details and thumb image is delivered to Dharmu over XMPP stream; and the Dharmu's device will display the file thumb with an option of downloading the file.

Once chosen to be downloaded, following operations will be executed-

- Negotiate a SFCM Connection with Server
- Send an IQ Packet to Server to send the file
- Receive the file on SFCM Connection

As a first step, device will negotiate a SFCM connection with the server (assuming there is no active SFCM connection). Once the connection has been negotiated, the device will send an IQ packet with file-id and connection SID over authenticated XMPP stream. The server validates the fileId and connection SID and if found valid, returns a success response.

Upon receiving the success response from Server, the device downloads the file from SFCM Connection.

Below is the IQ Packet sent to Sever to request file download from device-

```
<iq id='123' from='dharmu@alterbasics.com' to='alterbasics.com' type='get'>
    <query xmlns='stringflow:media'>
        <media media-id=' ftsksd-78snd-bsnd88-nsd9' sid='xyz-123' />
    </query>
</iq>
```

The server responds with one of the following two responses conditioned upon successful validation of fileId and SID sent in the IQ packet.

IQ Success Response

```
<iq id='123' from='alterbasics.com' to='dharmu@alterbasics.com' type='result'>
    <query xmlns='stringflow:media'>
        <media media-id=' ftsksd-78snd-bsnd88-nsd9' sid='xyz-123' />
    </query>
</iq>
```

IQ Failure Response

```
<iq id='123' from='alterbasics.com' to='dharmu@alterbasics.com' type='error'>
    <query xmlns='stringflow:media'>
        <media media-id=' ftsksd-78snd-bsnd88-nsd9' sid='xyz-123' />
    </query>
</iq>
```

Synchronous File Transfer

As a concept, synchronous file transfer peer to peer file transfer between two users. The file content are transferred from one user device another user device and Stringflow server acts as a relay. While the mechanism does not require server to store and manage user files, it mandates both user to be available at the same time.

The feature is not supported in Stringflow Version: 1.0.0