[SendBird] Voice Notes

Date: 2020-03-26
Prepared By: Solutions Engineering

Background

- SendBird's file upload service allows for any data files to be uploaded and stored in SendBird.
- SendBird's FILE messages store URL links to both internal and external data
- This document contains information regarding considerations for uploading voice recordings and handling them in Android, iOS or JavaScript.

File Upload Size Limitations:

- Note: If you upload an audio file directly to SendBird, a size limit is imposed per file depending on your plan. You can see your limit in the dashboard (default 5MB) and adjust the limit via our sales team.
- No file size limit is imposed if you send a file message with its URL since the file isn't uploaded to SendBird server.
- Should a file exceed the default limit then the following error is returned:
    - "Maximum file size for uploading is 25MB". Code = 400111 ”

Moderation

Consider that voice notes allow for un-moderated sound to be sent through your messaging service. Users will be able to bypass text moderation, domain moderation, and image moderation by communicating via voice notes.

User Experience Considerations

Recording sound

As recording sound is outside of the scope of SendBird, here are some considerations rather and a definitive guide to capturing a user's voice recording.

- Permissions
  - Make sure to request permission from the user to record their devices audio. Deepening on the platform this maybe getUserMedia() in Javascript, updating the info.plist in iOS or Changing the manifest in Android
- Recording
  - Consider providing the user with at least two buttons.
    - Provide a button to record audio.
      - Be sure to indicate recording is in progress.
      - Also provide a way for the audio recording to stop.
    - Provide a button for the user to playback their recording and also re-record if needed.
- Recording data
  - Consider how the device will buffer the recording data locally, and how you can capture that data when the recording is done for uploading to SendBird.
    - For Android
      - Record
      - Capture data for uploading - Read data from audio file.
    - For iOS
      - Record
      - Capture data for uploading - Read data from AVAudiRecorder file.
    - For JavaScript
      - Record and buffer to a blob for uploading
  -

Async nature of SendBird's FILE messages:

- Sending SendBird FILE messages with the actual file data takes more time the larger the audio file.
- However, when a FILE message is received it is received at the same speed as a regular user message.
- This is because a received FILE message contains a url link to the file and not the actual data.

UX implementation considerations for large files:

- For large audio file uploads perhaps in your UI provide a progress indicator.
    - SendBird provides progress handlers that increment from 0.0 to 1.0 as the multipart upload progresses.
        - iOS
        - Android
        - JavaScript
    - Once the upload is done the progress indicator will be 1.0
- Also, consider the server side processing time. Once SendBird's server side operations are complete the FILE message callback will return a successful message object or an error message.
- Here is the above suggested order of operations in Swift.

```
self.message = groupChannel?.sendFileMessage(with: params, progressHandler: {
    bytesSent, totalBytesSent, totalBytesExpectedToSend in
    let progress = CGFloat(totalBytesSent) / CGFloat(totalBytesExpectedToSend)
    self.progressView.isHidden = false
    if progress == 1.0 {
        self.activityIndicator.isHidden = false
        self.progressView.isHidden = true
    }
    self.progressView.progress = Float(progress)
}, completionHandler:  {(message, error) in
    guard error == nil else {
        print("File Failed To Upload")
        return
    }
    self.activityIndicator.isHidden = true
    print(message)
})
```

Loss of internet connection

- Once sendFileMessage() has been called the SendBird's SDK Connection Manager will...
    - ...automatically keep watch over the file upload.
    - ...watch for times when the internet goes off for short periods.
    - ... automatically continue to send the file when the internet returns.
    - ...reset the connection manager everytime the internet comes back,

- So long as the total disconnection is not longer than 45 seconds the SendBird SDK will continue to try to send the video file data.

    - This cycle will continue infinitely, except for the iOS SDK which will timeout and stop the upload after 5 minutes from when the first part of the file was successfully sent.

- If the Connection Manager's event handler fires onReconnectFailed() all attempts to upload the file will stop and the sendFileMessage callback will return the error message. "File upload timeout error"
    - From within the onReconnectFailed() handler consider indicating to the user their connection is lost and either automatically try calling sendbird.reconnect() or offer the user the option to manually call sendbird.reconnect().
    - Once reconnected consider offering the user the option to retry sending the FILE message.
- Should the user lock the screen or move the app into the background once a file upload has begun the file upload process will continue so long as there is an internet connection.

Oversize audio files

Consider the case where a user attempts to send a file message where the file is larger than the file size limit set in your SendBird's Application payment Plan. For voice notes it might be worth considering to limit the recording length.

- By default the maximum file size is 5mb.
- If your maximum file size is predetermined, consider calculating the file size before the user attempts to send the file. Warn the user if their video file is oversize.
- SendBird's server sends an error should the attempted file upload exceed the maximum allowed limit.
    - "Maximum file size for uploading is 25MB". Code = 400111 "
    - The warning message triggers once uploading is done, not before.
    - For iOS consider setting the `SBDOptions.setFileTransferTime(300)`
        - Doing this will prevent the device from timing out a large upload.
- Should your user wish to cancel the upload at any time they can do so using the cancelUploadingFileMessage().
    - iOS
    - Android
    - JavaScript

Uploading audio files using a SendBird FILE message

- Android

- iOS
- JavaScript

Audio FILE message object

Audio file message example object (Example includes the optional static thumbnails)

```
{
    "custom_type": "",
    "mentioned_users": [],
    "updated_at": 0,
    "is_removed": false,
    "user": {
        "nickname": "User2",
        "user_id": "User2",
        "profile_url": "https://static.sendbird.com/sample/profiles/profile_14_512px.png",
        "metadata": {
            "team": "yellow"
        }
    },
    "file": {
        "url": "https://file-ap-1.sendbird.com/1660bab2d4bf4ce182ef23e2802c9275.m4a",//Animated
        "data": "",
        "type": "audio/m4a",
        "name": "1660bab2d4bf4ce182ef23e2802c9275.gif",
        "size": 446121
    },
    "thumbnails": [],
    "type": "FILE",
    "created_at": 1583901367885,
    "req_id": "6D1893C9-6F72-42E0-85EC-8E0B0FA4FDEC",
    "mention_type": "users",
    "channel_url": "sendbird_group_channel_127670705_e38390fc564369ab27f85efb9d7bc413fa72a474",//Static
    "message_id": 3614558059,
    "require_auth": true
}
```

Handling audio files in messages

Consider how you will inform the user that they have a message which contains an audio file.

- Above there is a sample FILE message object. Note that the FILE message object contains a field `file.type`.
- Perhaps consider the following points about delivering the audio to the user.
    - The FILE message contains a link to an audio file.
    - Will you download the whole audio before playing it or will you start downloading and playing at the same time? Both options are possible.
    - Please utilize SendBird's samples in as a way to implement the handling of File messages with audio:
        - iOS

        - Android
        - JavaScript

- Android
    - Fetch FILE message
        - Or listen for messages via channel handler:
            - onMessageReceived()
    - Target the message's video URL
        - FileMessage.getUrl()
    - Implementation specific processes
        - Offer the user playback options
        - Start downloading the audio file
        - Play audio file as it downloads

- iOS
    - Fetch FILE message
        - Or listen for messages via channel delegate:
            - channel:didReceiveMessage)
    - Target the message's audio file URL
        - message.url
    - Implementation specific processes
        - Offer the user playback options
        - Start downloading the audio file
        - Play audio file as it downloads