Testing Samsung Applications

Applications that are used for testing:

* Samsung My Files
* Samsung Camera
* Samsung Gallery
* Samsung smart switch
* Samsung galaxy wearable
* Samsung quick share

Android used 3 types of libraries for transferring files:

We have often come across three types of libraries after reverse engineering the above applications.

* zip.ZipEntry;
* zip.GZIP
* Zip4j

ZIP is only an archive format where we can add multiple files into a zip. When we try to send multiple files the zipentry creates a zip with one file in it and keeps adding the files to the zip.

Whereas Gzip is a compression format. Not all Samsung apps are using GZIP compression. Here also GZIP works extremely well for the text files but when it comes to media, it doesn’t compress them at all or doing zero compression. There isn’t much to do when it comes to video and audio since they are already compressed formats.

the android isn’t doing any compressing on the images either.

How I can confirm this, whenever an image or video is transferred, I have not seen a change in its size. If image is compressed and decompressed, we can actually see the size difference like a decrease or increase in the file.

In the test case, I have written I have seen a loss in the image size after decompression but whereas in Samsung apps the image size is the same. This can confirm that there isn’t anything happening when it comes to the compression of these formats. Also, I haven’t noticed any other compression formats other than GZIP in the Samsung and android apps.

Maybe the reason could be android wants to keep the original size of the pictures and doesn’t want to lose any details over compression and decompression.

I have also come across Bitmaps. It is an image file format that can be used to create and store graphics in android. Bitmaps also have image compression but bitmaps only deal with the graphics and user interface in the application. It doesn’t deal with or take any part in file transmission.

There is no universal inbuilt application to transfer files between android devices.Files have to be transferred via a third-party app like shareit, xender. These apps will establish an FTP connection using WiFi Direct.

Wi-Fi Direct enables android mobile phones to create their own Wi-Fi networks without an internet connection. It can also be used to connect cameras, printers, PCs, and gaming without active internet.

An alternate would be using third-party file system apps like Amaze file manager, Solid Explorer, MiXplorer which support FTP but they require setting up the client and server manually. These apps can also be used to transfer to computers since they establish direct connections within the network. They also use the FTP protocol but it requires setting up the FTP server manually by giving the IP and port.

zip.ZipEntry:

zip.ZipEntry belongs to the .zip library which creates a zip file when it is called. It can or add remove files to an already created zip. When multiple files are requested for sending it creates a zip file with a single file in it and keeps adding files to this zip. I have not seen any code for compression which is using .ZipEntry class.

The size of the file doesn’t change after using the.ZipEntry class.

Here’s a test case for zip.ZipEntry which takes any type of files and merges into one single zip file and zipoutputstream. Here I have used only images but we can use any type of files.



/\*\* package com.java2novice.zip; \*\*/

import java.io.BufferedOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.zip.ZipEntry;

import java.util.zip.ZipOutputStream;

public class MyMultipleFileZip{

public void zipFiles(List<String> files){

FileOutputStream fos = null;

ZipOutputStream zipOut = null;

FileInputStream fis = null;

try {

fos = new FileOutputStream("C:/Users/saitu/Downloads/Java\_Practice/testing.zip");

zipOut = new ZipOutputStream(new BufferedOutputStream(fos));

for(String filePath:files){

File input = new File(filePath);

fis = new FileInputStream(input);

ZipEntry ze = new ZipEntry(input.getName());

System.out.println("Zipping the file: "+input.getName());

zipOut.putNextEntry(ze);

byte[] tmp = new byte[4\*1024];

int size = 0;

while((size = fis.read(tmp)) != -1){

zipOut.write(tmp, 0, size);

}

zipOut.flush();

fis.close();

}

zipOut.close();

System.out.println("Done... Zipped the files...");

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} finally{

try{

if(fos != null) fos.close();

} catch(Exception ex){

}

}

}

public static void main(String a[]){

MyMultipleFileZip mfe = new MyMultipleFileZip();

List<String> files = new ArrayList<String>();

files.add("C:/Users/saitu/Downloads/Java\_Practice/TestImage/cat1.jpeg");

files.add("C:/Users/saitu/Downloads/Java\_Practice/TestImage/cat2.jpeg");

files.add("C:/Users/saitu/Downloads/Java\_Practice/TestImage/cat3.png");

mfe.zipFiles(files);

}

}

zip.GZIP:

zip.Gzip belongs to the .zip library which compresses the byte code received from ByteArrayOutputStream. Whenever a file is sent through ByteArrayOutputStream it converts the file to byte code. The byte code is sent to GZIP class for compression. Although, .GZIP is only used for compression I have not seen any code that is using both zip.Gzip and zip.ZipEntry classes.

Also, I have noticed a reduction the in the image size when the program is run but while transferring in the android it’s not happening.

This is an example of zip.Gzip which compresses and decompresses the byte code of an image. Further implemented any file can be turned into byte code and can be compressed using GZIP.



// Java Program to Usage of GZIPInputStream

// via Showcasing Reading Data

// Importing required classes

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.util.Arrays;

import java.util.zip.DataFormatException;

import java.util.zip.GZIPInputStream;

import java.util.zip.GZIPOutputStream;

import java.awt.image.BufferedImage;

import java.io.File;

import javax.imageio.ImageIO;

// Main class

// GZIPInputStreamDemo

public class GzipImage {

// Main driver method

public static void main(String[] args)

throws DataFormatException, IOException

{

BufferedImage bImage = ImageIO.read(new File("sample.jpg"));

ByteArrayOutputStream bos = new ByteArrayOutputStream();

ImageIO.write(bImage, "jpg", bos );

byte [] input = bos.toByteArray();

// Print and display the message

System.out.println("Original Message length : "

+ input.length);

// Compress the bytes

ByteArrayOutputStream arrayOutputStream

= new ByteArrayOutputStream();

GZIPOutputStream outputStream

= new GZIPOutputStream(arrayOutputStream);

outputStream.write(input);

outputStream.close();

// Read and decompress the data

byte[] readBuffer = new byte[5000];

ByteArrayInputStream arrayInputStream

= new ByteArrayInputStream(

arrayOutputStream.toByteArray());

GZIPInputStream inputStream

= new GZIPInputStream(arrayInputStream);

int read = inputStream.read(readBuffer, 0,

readBuffer.length);

inputStream.close();

// Should hold the original (reconstructed) data

byte [] result = bos.toByteArray();

System.out.println("Original Message length : "

+ result.length);

ByteArrayInputStream inStreambj = new ByteArrayInputStream(result);

BufferedImage newImage = ImageIO.read(inStreambj);

ImageIO.write(newImage, "jpg", new File("sampleoutput.jpg"));

}

}

Zip4j:

Zip4j was started by Srikanth Reddy Lingala back in 2008/2009 when he realized the lack of support for the majority of zip format features in Java. Both compression and zipping can be done using this library.

Zip4j is only used in the Samsung smart switch and wasn’t found in any other applications.

Features:

* Create, Add, Extract, Update, and Remove files from a zip file
* Support for streams (ZipInputStream and ZipOutputStream)
* Read/Writpassword-protected zip files and streams
* Support for both AES and zip standard encryption methods
* Support for Zip64 format
* Store (No Compression) and Deflate compression method
* Create or extract files from split zip files (Ex: z01, z02,...zip)
* Support for Unicode file names and comments in zip
* Progress Monitor - for integration into apps and user-facing applications

Here’s the code for file transfer between client and server.

**Clint side :**

import java.io.\*;

import java.net.Socket;

public class Client {

private static DataOutputStream dataOutputStream = null;

private static DataInputStream dataInputStream = null;

public static void main(String[] args)

{

// Create Client Socket connect to port 900

try (Socket socket = new Socket("localhost", 900)) {

dataInputStream = new DataInputStream(

socket.getInputStream());

dataOutputStream = new DataOutputStream(

socket.getOutputStream());

System.out.println(

"Sending the File to the Server");

// Call SendFile Method

sendFile(

"/home/dachman/Desktop/Program/gfg/JAVA\_Program/File Transfer/txt.pdf");

dataInputStream.close();

dataInputStream.close();

}

catch (Exception e) {

e.printStackTrace();

}

}

// sendFile function define here

private static void sendFile(String path)

throws Exception

{

int bytes = 0;

// Open the File where he located in your pc

File file = new File(path);

FileInputStream fileInputStream

= new FileInputStream(file);

// Here we send the File to Server

dataOutputStream.writeLong(file.length());

// Here we break file into chunks

byte[] buffer = new byte[4 \* 1024];

while ((bytes = fileInputStream.read(buffer))

!= -1) {

// Send the file to Server Socket

dataOutputStream.write(buffer, 0, bytes);

dataOutputStream.flush();

}

// close the file here

fileInputStream.close();

}

}

**Server Side:**

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.FileOutputStream;

import java.net.ServerSocket;

import java.net.Socket;

public class Server {

private static DataOutputStream dataOutputStream = null;

private static DataInputStream dataInputStream = null;

public static void main(String[] args)

{

// Here we define Server Socket running on port 900

try (ServerSocket serverSocket

= new ServerSocket(900)) {

System.out.println(

"Server is Starting in Port 900");

// Accept the Client request using accept method

Socket clientSocket = serverSocket.accept();

System.out.println("Connected");

dataInputStream = new DataInputStream(

clientSocket.getInputStream());

dataOutputStream = new DataOutputStream(

clientSocket.getOutputStream());

// Here we call receiveFile define new for that

// file

receiveFile("NewFile1.pdf");

dataInputStream.close();

dataOutputStream.close();

clientSocket.close();

}

catch (Exception e) {

e.printStackTrace();

}

}

// receive file function is start here

private static void receiveFile(String fileName)

throws Exception

{

int bytes = 0;

FileOutputStream fileOutputStream

= new FileOutputStream(fileName);

long size

= dataInputStream.readLong(); // read file size

byte[] buffer = new byte[4 \* 1024];

while (size > 0

&& (bytes = dataInputStream.read(

buffer, 0,

(int)Math.min(buffer.length, size)))

!= -1) {

// Here we write the file using write method

fileOutputStream.write(buffer, 0, bytes);

size -= bytes; // read upto file size

}

// Here we received file

System.out.println("File is Received");

fileOutputStream.close();

}

}