**Android: Share contents over an email**

Hello Friends,

Here we are discussing about how to share a content over email in android. Before starting that, I would like to know you about the actions overview that we use to send an Intent for email.

1. **ACTION\_SENDTO:** This action command is required to send an email to someone specified by the data. By using this command, you can’t share an attachment with that email. Also, you cannot send mail to more than 1 email id.
2. **ACTION\_SEND:** This Intent action is required to send an email to someone specified by the data. Optionally, you can also set some standard extras for the intent: EXTRA\_EMAIL, EXTRA\_CC, EXTRA\_BCC, EXTRA\_SUBJECT. With this intent, you can send an attachment as well, but you cannot send more than 1 file as an attachment.
3. **ACTION\_SEND\_MULTIPLE:** This intent action is required to send multiple contents over an email. With this intent, you can send email to multiple user’s email id’s and this action allows you to send more than one attachment with an email.

There are two ways, you can share the data.

1. To specific email client:

If you want to share a data through specific email client like for eg. Gmail app, then you must use package specific name and extension ".gm" or "gmail". You first query package manager to get the ResolveInfo list of all the available email apps. Just check whether the above-mentioned package or extension name is matches with an any of the ResolveInfo object from that list. If it matches, then you need to set that one using setClassName() method of an Intent class.

1. To user chosen email clients:

If you want to allow user to share data to any of the all available email apps in your device, then you must first create an email intent and query package manager to get the ResolveInfo list of all the available email apps by using that intent. Once you got the list, then create an intent chooser using Intent.createChooser() method and call activity. startActivity() by passing that intent as an argument to this method.

There can be situation where user want to send multiple attachments. In that case, you must have to add WRITE\_EXTERNAL\_STORAGE permission in your AndroidManifest.xml and allow user to grant the access by showing permission popup above marshmallow device. With this, you also need to create and add FileProvider in AndroidManifest.xml. It is a special subclass of ContentProvider that facilitates secure sharing of files associated with an app by creating a content uri for a file. A content URI allows you to grant read and write access for that email client app using temporary access permissions. If you didn’t add, then any email app like Gmail, cannot be able to attach provided file with newly created email.

Now let’s see the how to share a content and file over an email:

1. Add below code in AndroidManifest.xml file.

|  |
| --- |
| <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  <uses-permission android:name="android.permission.INTERNET" />  <uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  <uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE" /  <provider  android:name="com.example.datasharing.ShareMailFileProvider"  android:authorities="${applicationId}.provider"  android:exported="false"  android:grantUriPermissions="true">  <meta-data  android:name="android.support.FILE\_PROVIDER\_PATHS"  android:resource="@xml/provider\_paths" />  </provider> |

1. Create one “xml” directory in the res folder and add provider\_paths.xml file in that directory.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <paths>    <external-path  name="external\_files"  path="." />    <cache-path  name="cache"  path="/" />  </paths> |

1. Here I have created one model to hold all the email related information.

|  |
| --- |
| import java.io.File;  import java.util.List;  /\*\*  \* Class to set the data which requires to compose and send email.  \*/  public class EmailData {  /\*\*  \* Holds the intent action type for sending mail.  \*/  private String sendMailAction;  /\*\*  \* Holds the content type.  \*/  private String contentType;  /\*\*  \* Holds the mail subject.  \*/  private String mailSubject;  /\*\*  \* Holds the mail body.  \*/  private String mailBody;  /\*\*  \* Holds the string array of email id's to whom we are sending an email.  \*/  private String[] emailIdsTo;  /\*\*  \* Holds the account domain extensions.  \*/  private PackageExtension accountDomainExtension;  /\*\*  \* Holds the email app chooser dialog title.  \*/  private String emailAppChooserDialogTitle;  /\*\*  \* Holds the list of files to be attached with that mail.  \*/  private List<File> files;  /\*\*  \* Holds all email id's needed to keep in cc.  \*/  private String[] emailIdsCc;  /\*\*  \* Holds all email id's needed to keep in cc.  \*/  private String[] emailIdsBcc;  /\*\*  \* Constructor to initialize the email parameters.  \* @param emailComposer EmailComposer object which holds all email data.  \*/  private EmailData(EmailComposer emailComposer) {  this.sendMailAction = emailComposer.sendMailAction;  this.contentType = emailComposer.contentType;  this.mailSubject = emailComposer.mailSubject;  this.mailBody = emailComposer.mailBody;  this.emailIdsTo = emailComposer.emailIdsTo;  this.accountDomainExtension = emailComposer.accountDomainExtensions;  this.emailAppChooserDialogTitle = emailComposer.emailAppChooserDialogTitle;  this.files = emailComposer.files;  this.emailIdsCc = emailComposer.emailIdsCc;  this.emailIdsBcc = emailComposer.emailIdsBcc;  }  /\*\*  \* Returns all account domain extensions which belongs to particular email client app.  \* @return PackageExtension object.  \*/  PackageExtension getAccountDomainExtensions() {  return accountDomainExtension;  }  /\*\*  \* Returns email app chooser dialog title.  \* @return email app chooser dialog title.  \*/  String getEmailAppChooserDialogTitle() {  return emailAppChooserDialogTitle;  }  /\*\*  \* Returns intent action required for sending an email.  \* @return Returns intent action.  \*/  String getSendMailAction() {  return sendMailAction;  }  /\*\*  \* Returns the content type.  \* @return Content type.  \*/  String getContentType() {  return contentType;  }  /\*\*  \* Returns the mail subject.  \* @return Mail subject.  \*/  String getMailSubject() {  return mailSubject;  }  /\*\*  \* Returns the mail body.  \* @return Mail body.  \*/  String getMailBody() {  return mailBody;  }  /\*\*  \* Returns String array of email ids to whom we are sending an email by keepin them in 'To' part.  \* @return Email id's.  \*/  String[] getEmailIdsTo() {  return emailIdsTo;  }  /\*\*  \* Returns all the files to be attached with the mail.  \* @return List of files  \*/  List<File> getAllFilesToBeAttached() {  return files;  }  /\*\*  \* Returns all the email id's which has to be keep as cc.  \* @return String array of email id's.  \*/  String[] getEmailIdsCc() {  return emailIdsCc;  }  /\*\*  \* Returns all the email id's which has to be keep as bcc.  \* @return String array of email id's.  \*/  String[] getEmailIdsBcc() {  return emailIdsBcc;  }  /\*\*  \* Builder class.  \*/  public static class EmailComposer {  private String sendMailAction;  private String contentType;  private String mailSubject;  private String mailBody;  private String[] emailIdsTo;  private PackageExtension accountDomainExtensions;  private String emailAppChooserDialogTitle;  private List<File> files;  private String[] emailIdsCc;  private String[] emailIdsBcc;  /\*\*  \* Constructor to set parameters.  \* @param sendMailAction intent action.  \* @param contentType content type.  \* @param mailSubject mail subject text.  \* @param mailBody mail body text.  \* @param emailIdsTo email id to whom we have to send mail.  \*/  public EmailComposer(String sendMailAction, String contentType, String mailSubject, String mailBody, String[] emailIdsTo) {  this.sendMailAction = sendMailAction;  this.contentType = contentType;  this.mailSubject = mailSubject;  this.mailBody = mailBody;  this.emailIdsTo = emailIdsTo;  }  /\*\*  \* Sets the account specific domain extensions.  \* @param accountDomainExtensions PackageExtension object.  \* @return EmailComposer object.  \*/  public EmailComposer setAccountSpecificDomainExtensions(PackageExtension accountDomainExtensions) {  this.accountDomainExtensions = accountDomainExtensions;  return this;  }  /\*\*  \* Sets the email app chooser dialog title.  \* @param emailAppChooserDialogTitle Title for email app chooser dialog.  \* @return EmailComposer object.  \*/  public EmailComposer setEmailAppChooserDialogTitle(String emailAppChooserDialogTitle) {  this.emailAppChooserDialogTitle = emailAppChooserDialogTitle;  return this;  }  /\*\*  \* Sets the files to be attached.  \* @param files File list.  \* @return EmailComposer object.  \*/  public EmailComposer addFileAttachments(List<File> files) {  this.files = files;  return this;  }  /\*\*  \* Sets the cc email id's.  \* @param emailIdsCc cc email id's.  \* @return EmailComposer object.  \*/  public EmailComposer setEmailIdsCc(String[] emailIdsCc) {  this.emailIdsCc = emailIdsCc;  return this;  }  /\*\*  \* Sets the bcc email id's.  \* @param emailIdsBcc bcc email id's.  \* @return EmailComposer object.  \*/  public EmailComposer setEmailIdsBcc(String[] emailIdsBcc) {  this.emailIdsBcc = emailIdsBcc;  return this;  }  /\*\*  \* Composes the email.  \* @return EmailData object.  \*/  public EmailData compose() {  return new EmailData(this);  }  }  } |

1. Use below code to create an EmailData object :

|  |
| --- |
| EmailData.EmailComposer emailComposer = **new** EmailData.EmailComposer(Intent.***ACTION\_SEND***, **"text/plain"**, getString(R.string.***diagnostic\_mail\_subject***), getString(R.string.***body\_text***), **new** String[]{emailAddressTo}); emailComposer.setEmailAppChooserDialogTitle(**"share data via"**);  //create file object having data and add that file in the below list. List<File> files = **new** ArrayList<>(); files.add(file); emailComposer.addFileAttachments(files); emailComposer.setAccountSpecificDomainExtensions(**new** PackageExtension(**".gm"**,**"gmail"**)); **return** emailComposer.compose(); |

1. Use below methods to share the data over email:

|  |
| --- |
| */\*\*  \* Sends the data through user chosen email client.  \*  \** ***@param activity*** *Activity context.  \** ***@param emailData*** *EmailData object.  \** ***@param emailsClients*** *List of email apps ResolveInfo.  \** ***@param emailIntent*** *Email Intent.  \*/* **private static void** shareDataThroughUserChosenEmailClient(Activity activity, EmailData emailData, List<ResolveInfo> emailsClients, Intent emailIntent) {  List<Intent> intents = **new** ArrayList<>();  **for** (ResolveInfo resolveInfo : emailsClients) {  Intent target = **new** Intent(emailIntent);  target.setPackage(resolveInfo.**activityInfo**.**packageName**);  intents.add(target);  }  **if** (!intents.isEmpty()) {  Intent chooserIntent = Intent.*createChooser*(intents.remove(0),  emailData.getEmailAppChooserDialogTitle());  chooserIntent.putExtra(Intent.***EXTRA\_INITIAL\_INTENTS***,  intents.toArray(**new** Parcelable[intents.size()]));  activity.startActivity(chooserIntent);  } } |
| */\*\*  \* Sends the data through email client.  \*  \** ***@param activity*** *Activity context.  \** ***@param emailData*** *EmailData object to prepare email intent.  \** ***@throws*** *ShareDataException Throws when data is not proper.  \*/* **private static void** shareDataThroughEmailClient(Activity activity, EmailData emailData) **throws** ShareDataException {  *checkValidations*(activity, emailData);  Intent emailIntent = *createEmailIntent*(activity, emailData);  **final** List<ResolveInfo> emailsClients = *getAllEmailApps*(activity, emailIntent);  **if** (emailsClients.isEmpty()) {  Toast.*makeText*(activity, **"There are no email clients installed. Please install the email client."**, Toast.***LENGTH\_LONG***).show();  **return**;  }  PackageExtension packageExtension = emailData.getAccountDomainExtensions();  **if**(packageExtension != **null**) {  *shareDataThroughSpecificEmailClient*(activity, emailsClients, packageExtension, emailIntent);  } **else** {  *shareDataThroughUserChosenEmailClient*(activity, emailData, emailsClients, emailIntent);  } } |
| */\*\*  \* Sends the data through specific email client as per user chosen email app package extensions.  \*  \** ***@param activity*** *Activity context.  \** ***@param emailsClients*** *List of email apps ResolveInfo.  \** ***@param packageExtension*** *{****@link*** *PackageExtension} object.  \** ***@param emailIntent*** *Email Intent.  \** ***@throws*** *ShareDataException Throws when data is not proper.  \*/* **private static void** shareDataThroughSpecificEmailClient(Activity activity, List<ResolveInfo> emailsClients, PackageExtension packageExtension, Intent emailIntent) **throws** ShareDataException {  ResolveInfo resolveInfo = **null**;  **for** (**final** ResolveInfo info : emailsClients) {  **if** (info.**activityInfo**.**packageName**.endsWith(packageExtension.getPackageName()) || info.**activityInfo**.**name**.contains(packageExtension.getName())) {  resolveInfo = info;  **break**;  }  }  **if** (resolveInfo != **null**) {  emailIntent.setClassName(resolveInfo.**activityInfo**.**packageName**, resolveInfo.**activityInfo**.**name**);  } **else** {  **throw new** ShareDataException(**"No app available which matches with given app package details."**);  }  activity.startActivity(emailIntent);  } |
| */\*\*  \* Sets the data from {****@link*** *EmailData} object to intent and returns the same.  \*  \** ***@param emailDataObject*** *EmailData object.  \** ***@return*** *Intent.  \** ***@throws*** *ShareDataException Throws when data is not proper.  \*/* **private static** Intent createEmailIntent(Activity activity, EmailData emailDataObject) **throws** ShareDataException {  String action = emailDataObject.getSendMailAction();  **if**(!(action.equals(Intent.***ACTION\_SENDTO***) || action.equals(Intent.***ACTION\_SEND***) || action.equals(Intent.***ACTION\_SEND\_MULTIPLE***))) {  **throw new** ShareDataException(**"Invalid intent action. Please send action as ACTION\_SEND, ACTION\_SEND\_MULTIPLE or ACTION\_SENDTO."**);  }  Intent emailIntent = **new** Intent(action);  emailIntent.setData(Uri.*parse*(***MAIL\_TO***));  emailIntent.setType(emailDataObject.getContentType());  List <File> files = emailDataObject.getAllFilesToBeAttached();  **if**(action.equals(Intent.***ACTION\_SENDTO***)) {  **if**(files == **null** || files.isEmpty()) {  **if**(emailDataObject.getEmailIdsTo().**length** > 1) {  **throw new** ShareDataException(**"You cannot send mail to more than 1 email id with ACTION\_SENDTO."**);  } **else** {  emailIntent.setData(Uri.*parse*(***MAIL\_TO*** + emailDataObject.getEmailIdsTo()[0]));  }  } **else** {  **throw new** ShareDataException(**"You cannot send attachment with ACTION\_SENDTO."**);  }  } **else** {  emailIntent.putExtra(Intent.***EXTRA\_EMAIL***, emailDataObject.getEmailIdsTo());  emailIntent.putExtra(android.content.Intent.***EXTRA\_CC***, emailDataObject.getEmailIdsCc());  emailIntent.putExtra(android.content.Intent.***EXTRA\_BCC***, emailDataObject.getEmailIdsBcc());  **if** (action.equals(Intent.***ACTION\_SEND***) && files != **null** && !files.isEmpty() && files.size() > 1) {  **throw new** ShareDataException(**"You cannot send more than 1 attachment with ACTION\_SEND."**);  }  }  ArrayList <Uri> uriList = **new** ArrayList<>();  **if** (files != **null** && !files.isEmpty()) {  **for**(File file : files) {  **if** (file.exists()) {  uriList.add(ShareMailFileProvider.*getUriForFile*(activity, activity.getApplicationContext().getPackageName() + **".provider"**, file));  } **else** {  Toast.*makeText*(activity, **"File "** + file.getName() + **"doesn't exists in the device."**,Toast.***LENGTH\_SHORT***).show();  }  }  **if**(!uriList.isEmpty()) {  **if**(action.equals(Intent.***ACTION\_SENDTO***) || action.equals(Intent.***ACTION\_SEND***)) {  emailIntent.putExtra(Intent.***EXTRA\_STREAM***, uriList.get(0));  } **else** {  emailIntent.putParcelableArrayListExtra(Intent.***EXTRA\_STREAM***, uriList);  }  emailIntent.addFlags(Intent.***FLAG\_GRANT\_READ\_URI\_PERMISSION***);  }  }  emailIntent.putExtra(android.content.Intent.***EXTRA\_SUBJECT***, emailDataObject.getMailSubject());  emailIntent.putExtra(android.content.Intent.***EXTRA\_TEXT***, emailDataObject.getMailBody());  **return** emailIntent; } |
| */\*\*  \* Queries package manager to get the ResolveInfo list of all the available email apps.  \*  \** ***@param activity*** *Activity context.  \** ***@param intent*** *Intent to send the data.  \** ***@return*** *List of ResolveInfo object.  \*/* **private static** List<ResolveInfo> getAllEmailApps(Activity activity, Intent intent) {  **return** activity.getPackageManager().queryIntentActivities(intent, 0); }  */\*\*  \* Validates the data.  \*  \** ***@param activity*** *Activity context.  \** ***@param emailData*** *EmailData object.  \** ***@throws*** *ShareDataException Throws when data is not proper.  \*/* **private static void** checkValidations(Activity activity, EmailData emailData) **throws** ShareDataException {  **if** (activity == **null**) {  **throw new** ShareDataException(**"Activity context should not be null."**);  }  **if** (emailData == **null**) {  **throw new** ShareDataException(**"EmailData object should not be null."**);  } } |

I hope this tutorial will help you to share the data and multiple files over an email.