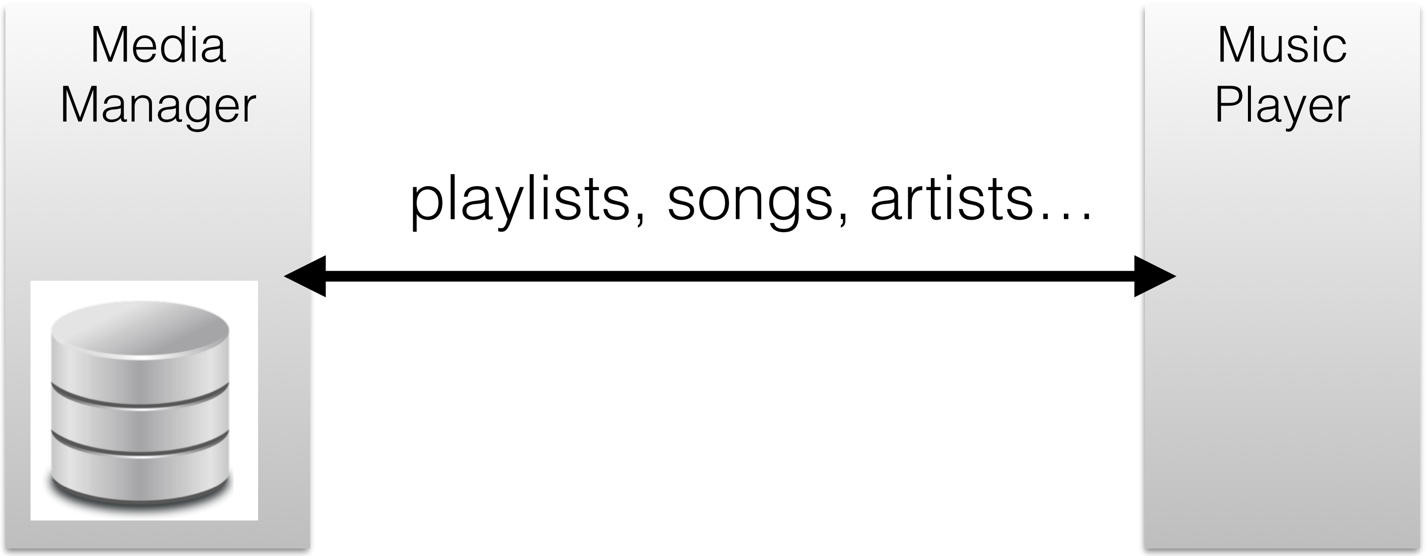
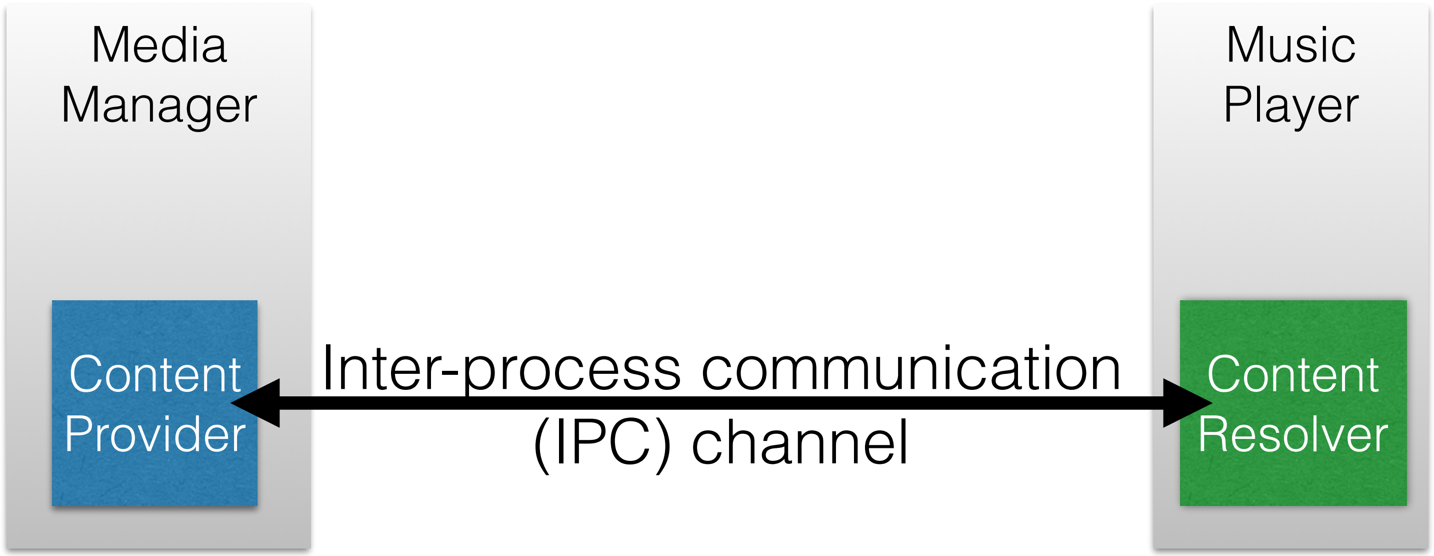
Android Content Providers

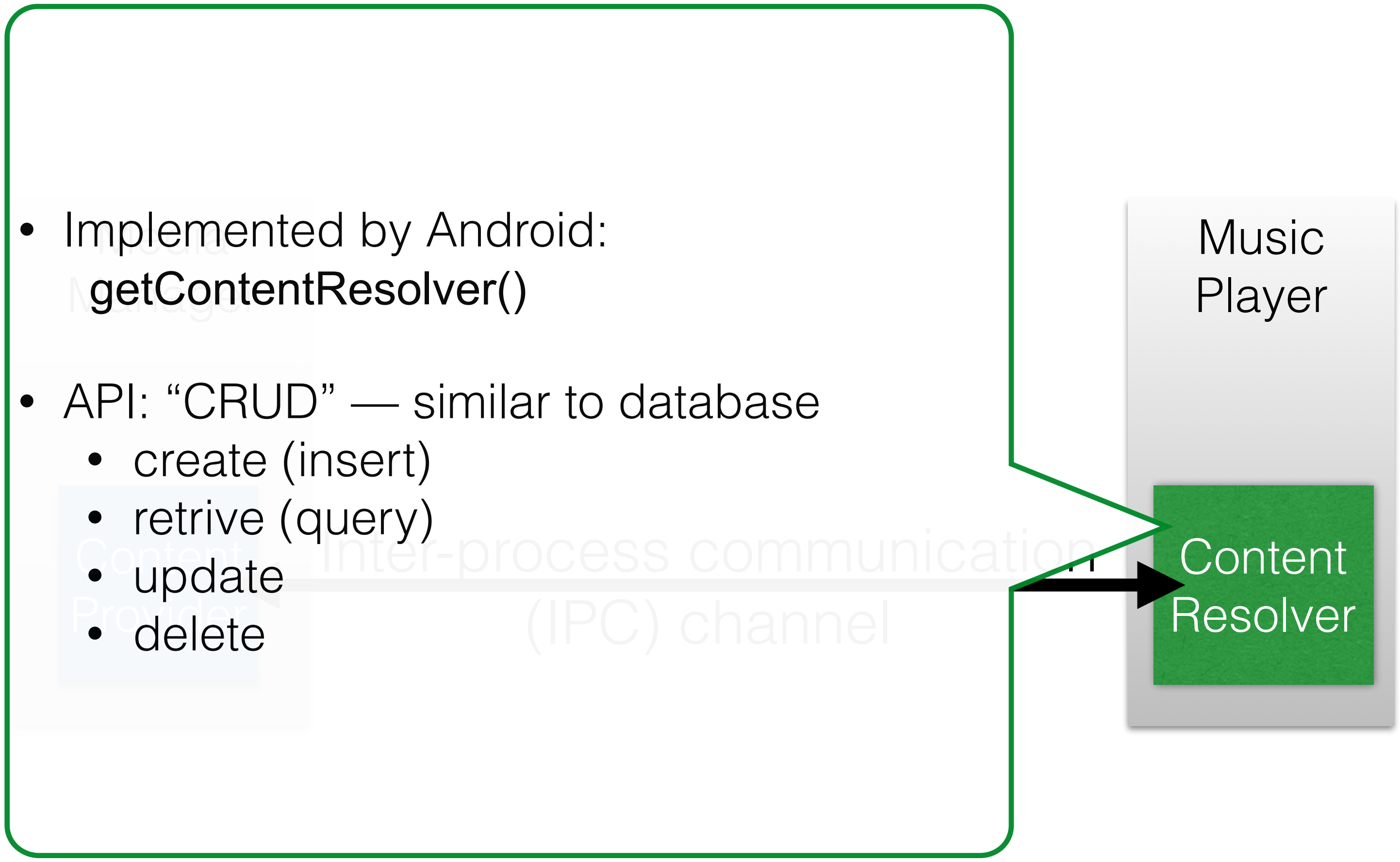
Using Media Data



Using Media Data



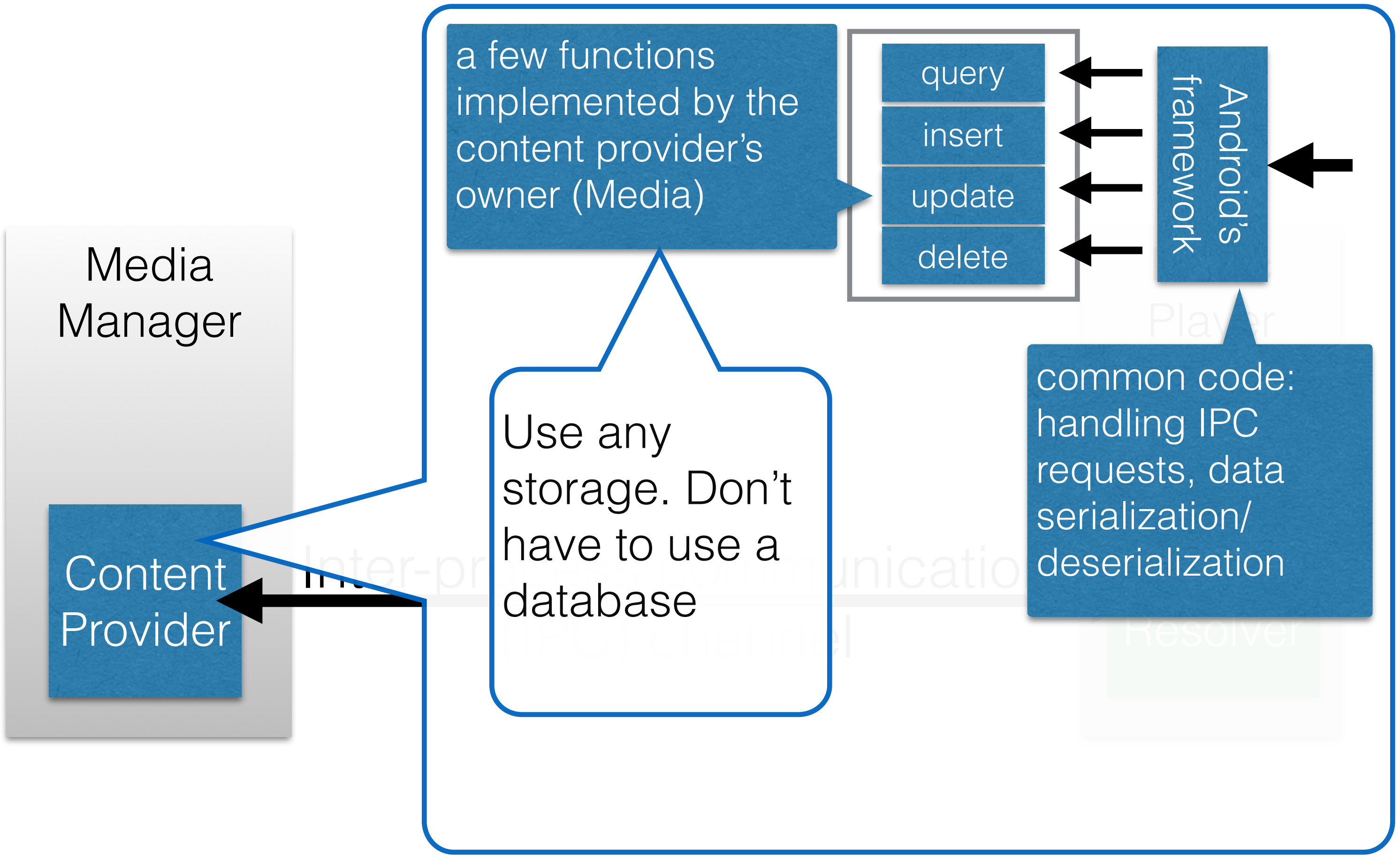
Client-side: content resolver



- Implemented by Android: getContentResolver()
- API: “CRUD” — similar to database
  - create (insert)
  - retrive (query)
  - update
  - delete

http://developer.android.com/reference/android/content/ContentResolver.html

Service-side: content provider


Media Manager
Content Provider
a few functions implemented by the content provider's owner (Media)
query
insert
update
delete
Android's framework
common code: handling IPC requests, data serialization/ deserialization
Use any storage. Don't have to use a database

Built-in content providers

- Contacts
- Media
- Calendar
- User Dictionary
- …

Simple example: user dictionary (built-in)

- Stores the spellings of non-standard words that the user wants to keepl

- Backed by a database table

word	app id	frequency	locale	_ID
mapreduce	user1	100	en_US	1
precompiler	user14	200	fr_FR	2
applet	user2	225	fr_CA	3
const	user1	255	pt_BR	4
int	user5	100	en_UK	5

Query from another app

get the ContentResolver object

```
mCursor = getContentResolver().query(
    UserDictionary.Words.CONTENT_URI,    // The content URI of the words table
    mProjection,                         // The columns to return for each row
    mSelectionClause                     // Selection criteria
    mSelectionArgs,                      // Selection criteria
    mSortOrder);                         // The sort order for the returned rows
```

URI: an identifier to locate the user dictionary

Locating resources using Content URIs

- scheme - always "content"
- authority - name of entire provider

} used by Android to identify a content provider

- path (optional)
  - data type path
  - instance identifier

} used by the content provider to identify **internal** objects

$$\underbrace{\texttt{content}}_{\text{scheme must be “content”}}\texttt{://}\underbrace{\texttt{user\_dictionary}}_{\text{authority}}\texttt{/}\overbrace{\texttt{words/5}}^{\text{Path}}$$

*For non-built-in apps: com.example.<appname>.provider

Uri class

- Convert String to Uri via Uri.parse()
  - Example:
    `Uri.parse("content://contacts/people");`

Creating a content provider

- Why?
  - You want to offer complex data or files to other applications.
  - You want to allow users to copy complex data from your app into other apps.
  - You want to provide custom search suggestions using the search framework.

Creating a content provider

- Design URI-to-data mapping
- Manifest declaration
- Implementation
- Permissions

http://developer.android.com/guide/topics/providers/content-provider-creating.html

Design URI-to-data mapping

- authority: user_dictionary
- path:
  - /words: all words
  - /words/<id>: a specific word
- Use **UriMatcher**

```
sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
sUriMatcher.addURI(AUTHORITY, "words", WORDS);
sUriMatcher.addURI(AUTHORITY, "words/#", WORD_ID);
```

http://androidxref.com/4.4.3_r1.1/xref/packages/providers/UserDictionaryProvider/src/com/android/providers/userdictionary/UserDictionaryProvider.java

Declare in manifest

A content provider is an app component

http://developer.android.com/guide/topics/manifest/provider-element.html

```
</application>
   …
   <!-- The Content Provider is declared -->
   <provider android:name="UserDictionaryProvider"
     android:authorities=“user_dictionary"
     android:syncable="false"
     android:multiprocess="false"
     android:exported="true"
     android:readPermission="android.permission.READ_USER_DICTIONARY"
     android:writePermission="android.permission.WRITE_USER_DICTIONARY" />
</application>
```

http://androidxref.com/4.4.3_r1.1/xref/packages/providers/UserDictionaryProvider/AndroidManifest.xml

Implementation

Implement a class that extends ContentProvider

```
public class UserDictionaryProvider extends ContentProvider
{
  insert(…);
  query(…);
  update(…);
  delete(…);
  …
}
```

Implementing query

Implement this function:

```
public Cursor query(
     Uri uri, String[] projection,
     String selection, String[] selectionArgs,
     String sortOrder);
```

Match Uri

content://user_dictionary/**words/1**

path segments: [“words”, “1”]

```
switch (sUriMatcher.mat
case WORDS:
    qb.setTables(USERDI
    qb.setProjectionMap
    break;
case WORD_ID:
    qb.setTables(USERDICT_TABLE_NA E);
    qb.setProjectionMap(sDictProje tionMap);
    qb.appendWhere(
        "_id" + "=" + uri.getPathSegments().get(1));
    break;
default:
    throw new IllegalArgumentException(
        "Unknown URI " + uri);
}
```

Query DB, then return cursor

Register a **ContentObserver**

Allow Android’s “CursorLoader” mechanism to automatically re-fetch data

```
//                                                        t
Str
if

} e

}

// Get th  database and run the query
SQLiteDat base db = mOpenHelper.getReadableDatabase();
Cursor c   qb.query(db, projection, selection,
      sel ctionArgs, null, null, orderBy);

// Tell the cursor what uri to watch, so it knows when its
source data changes
c.setNotificationUri(
     getContext().getContentResolver(), uri);

return c;
```

Implementing insert

```
@Override
public Uri insert(Uri uri, ContentValues initialValues) {
    // Validate the requested uri
    if (sUriMatcher.match(uri) != WORDS) {
        throw new I[illegible]RI " + uri);
    }
    ContentValues v[illegible]
     … // sanitize initialValues [illegible]nd store to values

    SQLiteDatabase db = mOpenHelper.getWritableDatabase();
    long rowId = db.insert(
       USERDICT_TABLE_NAME, Words.WORD, values);
    if (rowId > 0) {
        Uri wordUri = ContentUris.withAppendedId(
                 UserDictionary.Words.CONTENT_URI, rowId);
        getContext().getContentResolver().notifyChange(
                 wordUri, null);
        mBackupManager.dataChanged();
        return wordUri;
    }
    throw new SQLException("Failed to insert row into " + uri);
}
```

return the inserted URI

notify content observers

Permissions in manifest

http://develo ider-element.html

exported: enable to share with other apps

```
</applicatio
   …
   <!-- The
   <provider android: ame="UserDictionaryProvider"
     android:authorit es="user_dictionary"
     android:syncable ="false"
     android:multiprocess="false"
     android:exported="true"
     android:readPermission="android.permission.READ_USER_DICTIONARY"
     android:writePermission="android.permission.WRITE_USER_DICTIONARY" />
</application>
```

http://androidxref.com/4.4.3_r1.1/xref/packages/providers/UserDictionaryProvider/AndroidManifest.xml

Permissions in manifest

http://developer.a   element.html

```
</application>
   …
   <!-- The Conte
   <provider andr
     android:authorities="  ser_dictionary"
     android:syncable="fal  e"
     android:multiprocess=  false"
     android:exported="tru  "
     android:readPermission="android.permission.READ_USER_DICTIONARY"
     android:writePermission="android.permission.WRITE_USER_DICTIONARY" />
</application>
```

read/write permissions

http://androidxref.com/4.4.3_r1.1/xref/packages/providers/UserDictionaryProvider/AndroidManifest.xml

Permissions on whole content provider

- Single read-write provider-level permission
  - One permission that controls both read and write access to the entire provider, specified with the android:permission attribute of the <provider> element.
- Separate read and write provider-level permission
  - You specify them with the android:readPermission and android:writePermission attributes of the <provider> element. They take precedence over the permission required by android:permission.

Path-level permissions

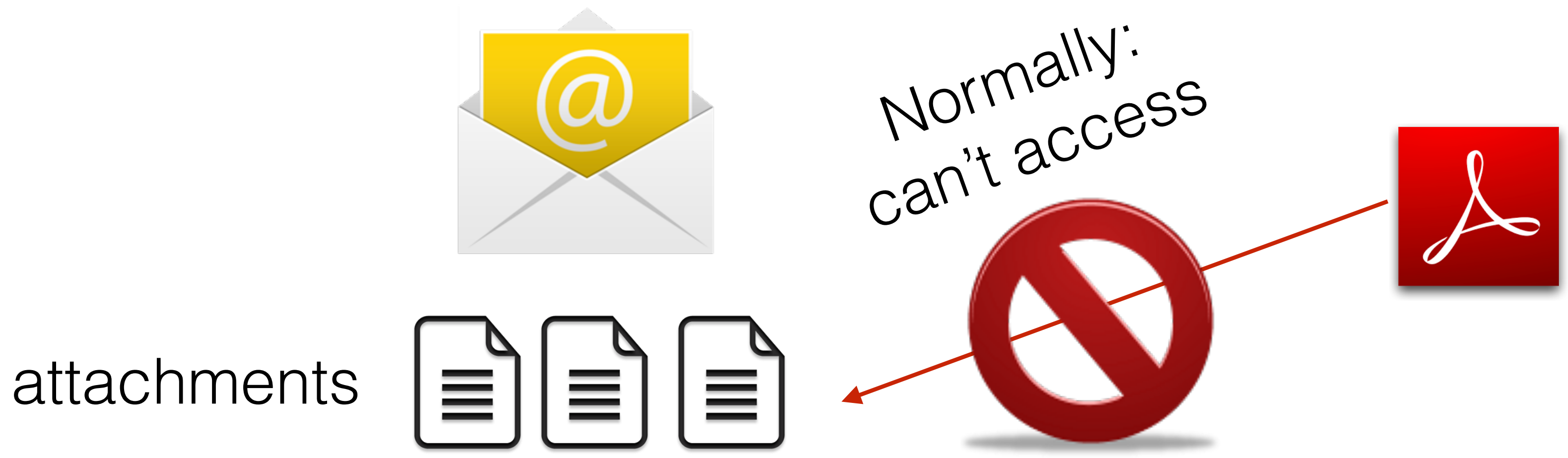
You specify **each URI** with a <path-permission> child element of the <provider> element. For each content URI you specify, you can specify a read/write permission, a read permission, or a write permission, or all three.

Path-level permission takes precedence over provider-level permissions.

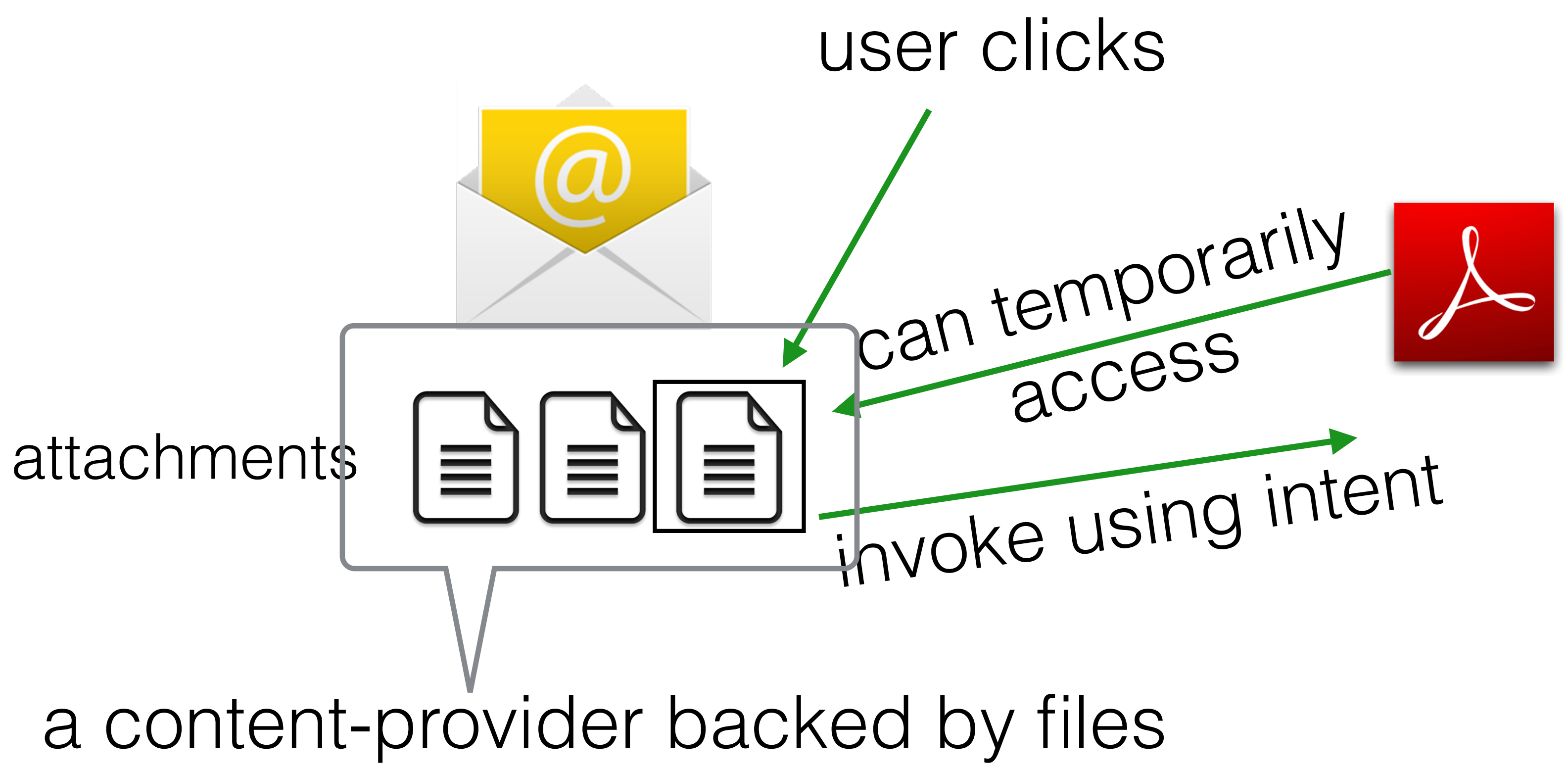
Temporary permissions

- Temporarily grant an app access
  - In the context of an invocation using an **intent**. — revoked when this invocation ends.
  - To a **specific URI** specified in the intent.

Example: email attachments



Example: email attachments



Temporary permissions

- Manifest: assert android:grantUriPermissions attribute in the <provider> element.
  - The scope of these permissions can be further limited by the <grant-uri-permission>.
- Intent (runtime): using the FLAG_GRANT_READ_URI_PERMISSION and FLAG_GRANT_WRITE_URI_PERMISSION flags in the Intent object that activates the component.

Example: email attachments

Invoke using intent

```java
/**
 * Returns an <code>Intent</code> to load the given attachment.
 * @param context the caller's context
 * @param accountId the account associated with the attachment (or 0 if we don't need to
 *     resolve from attachmentUri to contentUri)
 * @return an Intent suitable for viewing the attachment
 */
public Intent getAttachmentIntent(Context context, long accountId) {
    Uri contentUri = getUriForIntent(context, accountId);
    Intent intent = new Intent(Intent.ACTION_VIEW);
    intent.setDataAndType(contentUri, mContentType);
    intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION
            | Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
    return intent;
}

protected Uri getUriForIntent(Context context, long accountId) {
    Uri contentUri = AttachmentUtilities.getAttachmentUri(accountId, mId);
    if (accountId > 0) {
        contentUri = AttachmentUtilities.resolveAttachmentIdToContentUri(
                context.getContentResolver(), contentUri);
    }

    return contentUri;
}
```

Example: email attachments

Enable in manifest

```
<provider
        android:authorities="@string/eml_attachment_provider"
        android:exported="false"
        android:name="com.android.mail.providers.EmlAttachmentProvider" >
    <grant-uri-permission android:pathPattern=".*" />
</provider>
```

Example: email attachments

Implement file-related function in ContentProvider

```
public ParcelFileDescriptor openFile(
    Uri uri, String mode)
  throws FileNotFoundException
```

http://androidxref.com/4.4.3_r1.1/xref/packages/apps/Email/src/com/android/email/provider/AttachmentProvider.java

Android's built-in file-backed content provider class

- FileProvider: a subclass of ContentProvider
  - Implemented by Android
  - Supports simple filename-to-URI mapping

https://developer.android.com/reference/android/support/v4/content/FileProvider.html