Introduction

Android can access data through various built-in Content Providers such as contacts, browser, call logs etc. Content providers manage access to a structured set of data. One such Content Provider is android.provider.CallLog.Calls which provides access to the call logs data. Call Logs contain information about outgoing, incoming and missed calls.

Accessing Call Logs

The following steps show how to access the call log history: 1. Get the URI of the call log content provider. 2. Query the content provider for specific information. 3. Read data through the use of Cursor.

1. Get the URI of the Call Log Content Provider

android.provider.CallLog.Calls is a static class that exposes a CONTENT_URI field for the call log content provider (database). The CONTENT_URI field indicates the table from where the call logs data would be access. CONTENT_URI will be passed to the query method as explained in step 2.

2. Query the Content Provider for Specific Information.

The android.content.ContentResolver query() method is used to query the content provider.

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

Querying the given URI returns a Cursor over the result set. Parameters of the query(…) method are:

- uri - will be android.provider.CallLog.Calls (Where to get data from ?)
- projection - will be the columns to be return. Passing null will return all columns of the content provider (What columns to return?)
- sortOrder - the sorting order of the rows (How to sort the rows?.)

The following table presents the android.provider.CallLog.Calls columns constants that can be used for projections:

CACHED_NAME (String)	The cached name associated with the phone number, if it exists.
CACHED_NUMBER_LABEL (String)	The cached number label for a custom number type associated with the phone number, if it exists.
CACHED_NUMBER_TYPE (String)	The cached number type (Home, Work, etc) associated with the phone number, if it exists.
DATE (String)	The date the call occurred in milliseconds since an epoch
DURATION (String)	The duration of the call in seconds
IS_READ (String)	Whether an item is read or consumed by the user.
NEW (String)	Whether or not the call has been acknowledged
NUMBER (String)	The phone number as the user entered it.
TYPE	The type of the call (incoming, outgoing or missed).

(String)

The following code snippet shows the use of query method to access call logs from content provider.

```
ContentResolver cr = getContentResolver();
 /*method present in android.content.Contextclass*/
Cursor callLogCursor = cr.query(android.provider.CallLog.Calls.CONTENT_URI, /*uri*/
                                                                       null, /*projection*/
                                                                       null,/*selection*/
                                                                       null,/*selection
arguments*/

android.provider.CallLog.Calls.DEFAULT_SORT_ORDER /*sort by*/);
```

3. Read data through the use of android.database.Cursor

Cursor provides read access to the result set returned by the query() method. To read the data loop through the result set. Here call log information such as name of the contact, telephone number, date and time of call is accessed.

```
if (callLogCursor != null) {
         /*Looping through the results*/
         while (callLogCursor.moveToNext())
         {
                  /*Contact Name*/
                  String name =
callLogCursor.getString(callLogCursor.getColumnIndex(CallLog.Calls.CACHED_NAME));

                  String cacheNumber =
callLogCursor.getString(callLogCursor.getColumnIndex(CallLog.Calls.CACHED_NUMBER_LABEL));

                  /*Contact Number*/
                  String number =
callLogCursor.getString(CallLogCursor.getColumnIndex(CallLog.Calls.NUMBER));

                  /*Date*/
                  long dateTimeMillis =
callLogCursor.getLong(callLogCursor.getColumnIndex(CallLog.Calls.DATE));

                  /*Duration*/
                  long durationMillis =
callLogCursor.getLong(callLogCursor.getColumnIndex(CallLog.Calls.DURATION));

                  /*Call Type – Incoming, Outgoing, Missed*/
                  int callType =
callLogCursor.getInt(callLogCursor.getColumnIndex(CallLog.Calls.TYPE));
          }
```

```
        callLogCursor.close();
}
```

Call Logs Permission

To access call logs add the android.permission.READ_CONTACTS permission to your manifest.xml file.

```
<uses-permission android:name="android.permission.READ_CONTACTS">
```

Sample Example

NOTE: The following section explains about the important part of the application. Complete code snippet is provided at the end of the article as attachment.
The code sample consists of two activities namely CallLogDemoActivity and CallLogs activity class and two classes CallLogModel and CallLogsArrayAdapter class.

CallLogDemoActivity shows three buttons for outgoing, incoming and missed calls information access. The corresponding layout file is shown below

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">

    <Button
        android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Outgoing Call List" />
    <Button
        android:id="@+id/button2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Incoming Call List" />
    <Button
        android:id="@+id/button3"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Missed Call List" />

</LinearLayout>
```

Next, when the user clicks the corresponding button, the relevant data is accessed from the content provider and is presented to user in the form of list. The corresponding CallLogs Activity layout file is shown below. It consists of ListView.

listview.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <ListView
        android:id="@+id/listView"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >
    </ListView>
</LinearLayout>
```

Individual row of list is inflated using the below layout xml file. It consists of four TextView to show Name of the Contact, Number of the Contact, Date of Call and Duration of Call.

list_item.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/nameTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="15sp"/>
    <TextView
        android:id="@+id/numberTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="15sp"/>
    <TextView
        android:id="@+id/dateTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="15sp"/>
    <TextView
        android:id="@+id/durationTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="15sp"/>
</LinearLayout>
```

CallLogModel.java

CallLogModel class is created for storing the data information obtained from the call log Content Provider.

This Model class stores the data namely Name of the Contact, Number of the Contact, Duration of call and Date of call. The data is used in listview for displaying to the user.

```
public class CallLogModel {

         /*Name of the Contact*/
         private String name;
         /*Contact Number*/
         private String number;
         /*Duration of the call*/
         private String duration;
         /*Date of the call*/
         private String date;

         /*Constructor*/
         public CallLogModel(String name, String number, String duration, String date) {
          this.name = name;
          this.number = number;
          this.duration = duration;
          this.date = date;
         }

         /*Contact Name*/
         public String getName() {
          return name;
         }

         /*Contact Number*/
         public String getNumber() {
          return number;
         }

         /*Duration of call*/
         public String getDuration() {
          return duration;
         }

         /*Date of call*/
         public String getDate() {
          return date;
         }
}
```

CallLogsDemoActivity.java

Separate ArrayList is created for each call type as shown below.

```
ArrayList<CallLogModel> outgoingList = new ArrayList<CallLogModel>();
ArrayList<CallLogModel> incomingList = new ArrayList<CallLogModel>();
ArrayList<CallLogModel> missedcallList = new ArrayList<CallLogModel>();
```

On Application Launch, Activity calls the readCallLogs() method. readCallLogs() method queries ContentProvider and loops through the cursor. The CallLogModel object is created for each call type and is stored in the corresponding arraylist.

```
private void readCallLogs() {
         missedcallList.clear();
         incomingList.clear();
         outgoingList.clear();

         /*Query Call Log Content Provider*/
         Cursor callLogCursor =
getContentResolver().query(android.provider.CallLog.Calls.CONTENT_URI, null, null, null,
android.provider.CallLog.Calls.DEFAULT_SORT_ORDER);

         /*Check if cursor is not null*/
         if (callLogCursor != null) {

         /*Loop through the cursor*/
         while (callLogCursor.moveToNext()) {

                  /*Get ID of call*/
                  String id =
callLogCursor.getString(callLogCursor.getColumnIndex(CallLog.Calls._ID));

                  /*Get Contact Name*/
                  String name =
callLogCursor.getString(callLogCursor.getColumnIndex(CallLog.Calls.CACHED_NAME));

                  /*Get Contact Cache Number*/
                  String cacheNumber =
callLogCursor.getString(callLogCursor.getColumnIndex(CallLog.Calls.CACHED_NUMBER_LABEL));

                  /*Get Contact Number*/
                  String number =
callLogCursor.getString(callLogCursor.getColumnIndex(CallLog.Calls.NUMBER));

                  /*Get Date and time information*/
                  long dateTimeMillis =
callLogCursor.getLong(callLogCursor.getColumnIndex(CallLog.Calls.DATE));
                  long durationMillis =
callLogCursor.getLong(callLogCursor.getColumnIndex(CallLog.Calls.DURATION));

                  /*Get Call Type*/
                  int callType =
callLogCursor.getInt(callLogCursor.getColumnIndex(CallLog.Calls.TYPE));

                  String duration = getDuration(durationMillis * 1000);

                  String dateString = getDateTime(dateTimeMillis);

                  if (cacheNumber == null)
                                    cacheNumber = number;

                  if (name == null)
                           name = "No Name";

                  /*Create Model Object*/
                  CallLogModel callLogModel = new CallLogModel(name, cacheNumber, duration,
dateString);

                  /*Add it into respective ArrayList*/
                  if (callType == CallLog.Calls.OUTGOING_TYPE) {
                           outgoingList.add(callLogModel);
                  } else if (callType == CallLog.Calls.INCOMING_TYPE) {
                           incomingList.add(callLogModel);
                  } else if (callType == CallLog.Calls.MISSED_TYPE) {
                           missedcallList.add(callLogModel);
                  }
         }

         /*Close the cursor*/
         callLogCursor.close();
  }
}
```

Next on click of corresponding button, the CallLogs Activity is started. For example as shown below, on click of missed call button, the showMissedList() method is called which in turn starts the CallLogs Activity.

```
/*Constants to identify call type */
public static final int OUTGOING_CALLS = 1;
public static final int INCOMING_CALLS = 2;
public static final int MISSED_CALLS = 3;

public static final String CALL_TYPE = "TYPE";

/*On Click of Missed call button*/
private void showMissedList() {
 Intent intent = new Intent(this, CallLogs.class);
 intent.putExtra(CALL_TYPE, MISSED_CALLS);
 startActivity(intent);
}
```

CallLogs.java

CallLogs activity shows the call information using list. Here based on the call type, listAdapter is created as shown below.

```
/*listview to show call logs information*/
private ListView mainListView;

/*list adapter for binding data to view*/
private ArrayAdapter listAdapter;

public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.listview);
         mainListView = (ListView) findViewById(R.id.listView);

         /*Get Bundle*/
         Bundle extras = getIntent().getExtras();

         /*Check for the call type*/
         int callType = extras.getInt(CallLogsDemoActivity.CALL_TYPE);

         /*Create corresponding ArrayAdapter */
         if (callType == CallLogsDemoActivity.OUTGOING_CALLS)
                  listAdapter = new CallLogsArrayAdapter(this,
                                    CallLogsDemoActivity.outgoingList);
         else if (callType == CallLogsDemoActivity.INCOMING_CALLS)
                  listAdapter = new CallLogsArrayAdapter(this,
                                    CallLogsDemoActivity.incomingList);
         else if (callType == CallLogsDemoActivity.MISSED_CALLS)
                  listAdapter = new CallLogsArrayAdapter(this,
                                    CallLogsDemoActivity.missedcallList);

         /*Set adapter to listview*/
```

```
        mainListView.setAdapter(listAdapter);
}
```

CallLogsArrayAdapter.java

The CallLogsArrayAdapter class is an adapter class for listview. As shown below in the getView() method, each individual row(view) is created and is shown to the user.

```
/*Adapter getView() method does the work of binding data to view*/
public View getView(int position, View convertView, ViewGroup parent) {

        final CallLogModel callLogModel = this.getItem(position);
        TextView nameTV;
        TextView numberTV;
        TextView dateTV;
        TextView durationTV;

        /*Code is kept simple and not optimized*/
        convertView = inflater.inflate(R.layout.list_item, null);

        /* Find the child views.*/
        nameTV = (TextView) convertView.findViewById(R.id.nameTV);
        numberTV = (TextView) convertView.findViewById(R.id.numberTV);
        dateTV = (TextView) convertView.findViewById(R.id.dateTV);
        durationTV = (TextView) convertView.findViewById(R.id.durationTV);

        /*Set the information to the widget*/
        nameTV.setText("Name: "+callLogModel.getName());
        numberTV.setText("Number: "+callLogModel.getNumber());
        dateTV.setText("Date & Time: "+callLogModel.getDate());
        durationTV.setText("Duration:"+callLogModel.getDuration()+" secs");

        return convertView;
}
```

ref:http://developer.samsung.com/android/technical-docs/CallLogs-in-Android