Technical Document

Markit Charts Android Integration Guide

Version: 1.0.0

1. Package Contents

MarkitCharts consists of the following files:

- Markit Charts Android Integration Guide (This file)
- MarkitCharts - Android Only Properties Properties that only pertain to the Android SDK
- MarkitCharts - Share Properties Property name and description
- SampleProject An Example application showing the use of MarkitCharts SDK
- MarkitCharts JavaDocs The Java compiled documents from the SDK
- markitcharts-2.0.0-beta-1.aar MarkitCharts Framework

2. Installation

2.1 Device and OS Support

The Markit Charts SDK has been tested in the following scenarios. Other combinations may work.

OS:
- Android 4.1+

Device Support
- Samsung Galaxy, Note, and Tab Series
- Nexus devices (Phones & Tablets)

2.2 Required Frameworks

To integrate MarkitCharts the following frameworks must be included in your application.

2.2.1 Adding MarkitCharts Framework

- Add this to your projects lib folder:
  - markitcharts-***[current version]***.aar
    - For example: (markitcharts-2.0.0-beta-1.aar)

2.2.2 Adding Shinobi Framework

1. Open Android Studio
2. Go to File -> Import Module
3. Select the directory and leave module name as is
4. Open the shinobichartsandroidlibrary gradle file from the distributed folder
5. Replace apply plugin: 'com.android.application' with apply plugin: 'com.android.library'
6. Comment out applicationId "com.shinobicontrols.charts"

2.2.3 Adding Android Dependencies

Below is the listed dependencies used in the Markit Chart's SDK.

2.2.3.1 Android imports

```
import markit.android.DataObjects.ChartDataRequest;
import markit.android.DataObjects.DrawingMode;
import markit.android.DataObjects.Frequency;
import markit.android.DataObjects.Indicator;
import markit.android.DataObjects.Indicators.Price;
import markit.android.DataObjects.MenuChoice;
import markit.android.DataObjects.ReturnsData;
import markit.android.DataObjects.TimeFrame;
import markit.android.Interfaces.ChartHandler;
import markit.android.Interfaces.ChartUpdates;
import markit.android.MarkitCharts;
```

***Note**: You may not use all imports but above is listed as imports used to construct what is visible in the sample app*

2.2.3.2 Gradle Dependencies

Open build.gradle file in your application and add the following dependencies

```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile(name: 'markitcharts-[current version]', ext: 'aar')
    compile project(':shinobichartsandroidlibrary')
}

repositories{
     flatDir{
          dirs 'libs'
     }
}
```

2.2.3.3 Third Party Dependencies

- The MarkitChart SDK uses Jackson for deserializing data into objects, so by including the MarkitChart SDK you will already have that dependency included. you may have to remove references to Jackson if you are already using it.

  At the time of this document we are using the following versions of Jackson

```
com.fasterxml.jackson.core:jackson-databind:2.7.3
com.fasterxml.jackson.core:jackson-core:2.7.3
com.fasterxml.jackson.core:jackson-annotations:2.7.3
```

3.0 Example

3.1 Adding MarkitCharts

The activity that will be listening to chart updates must implement the ChartUpdates interface to receive chart updates.

```
public class MainActivity extends FragmentActivity implements ChartUpdates

private boolean showDarkTheme = true;
  @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
       chartHandler = MarkitCharts.newInstance(getSupportFragmentManager(),
          R.id.chart, showDarkTheme);
}
```

- ChartHandler - Is the interface for manipulating the MarkitCharts SDK's properties.
- R.id.chart - This is your reference to what container layout you want to be replaced with the chart fragment
- showDarkTheme is a boolean representing which base theme should be enabled (true is dark theme, false is light theme)

3.2 Getting Chart Data

```
private void RetrieveChartData() {
        chartHandler.setMainIssueId(issueId)
        ChartDataRequest chartDataRequest = chartHandler.getChartDataRequest();
        JSONObject json = chartDataRequest.getJSONObject();
        String chartAPIUrl = chartDataRequest.getAPIUrl(context);

        // Do Data Request with supplied API url and post body json
}
```

Now you can use your network stack to post to the url from the chartAPIUrl string and posting the json to that url.

3.3 Drawing the Chart

Once the data has come back from the API that data result needs to be passed back to the charting framework. This can be accomplished easily as follows:

```
private void drawCharts(String json) {
        json = (result from data call in section 3.2)
        chartHandler.drawCharts(json)
}
```

The json can be either a String or a JSONObject.