

Android Consumer Application

deviceWISE
Application Enablement Platform



Version 1.0
September 2015

Table of Contents

Table of Contents.....	2
Introduction.....	4
Android Consumer Application	4
Prerequisites	4
Android Project Structure.....	4
Interface.....	8
Layouts:.....	8
Main:.....	8
List of Organizations:.....	9
List of Things & Thing details:	10
Thing Details:.....	10
Backend	11
Major files in the Android project:.....	11
The Android manifest file.....	11
Layout file.....	12
Activity class	12
Documents structure.....	13
Authentication – “Main” screen	14
api.authenticate.....	14
user.find.....	16
Organizations – “Organizations” screen.....	17
Api Call: Session.org.list.....	17
Thing List - “things” screen.....	20
Api Call: Thing.list.....	20
Thing Details – “thing_details” screen.....	24
API Call: Thing.find.....	24
Alarms – “thing_alarms” screen.....	27
Java Method: onCreate()	27
Java Method: auxFunction()	28
Java Method: apiProps()	28
Java Method: GetSearchResults()	29
Attributes – “thing_attributes” screen	30
Java Method: onCreate()	30

Java Method: auxFunction()	30
Java Method: apiAttrs()	31
Properties – “thing_properties” screen	32
Java Method: onCreate()	32
Java Method: auxFunction()	33
Java Method: apiAttrs()	34
Java Method: GetSearchResults()	35
Method – “thing_method” screen	35
Java Class: MethodDetail	35
Java Class: method	36
API Call: thind_def.find	36
Java Method: textArea()	42
Processing the Commands	42
Java Method: apiCall()	42
Mobile project	44

Introduction

The deviceWISE M2M Application Enablement Platform (AEP) provides an HTTP interface that enables programmers to create HTTP based applications that either use the TR50 endpoint API or the simple REST API.

In order to properly understand the context of the information in this document the Android Consumer Application source code file should be readily available for reference. Code snippets here are intended only for clarity and are not intended to fully explain the complete workings of the Android application.

Android Consumer Application

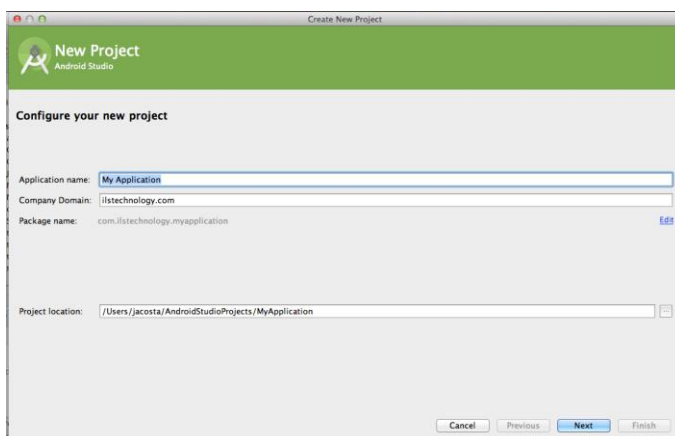
This Android consumer application interacts with deviceWISE M2M Service using JSON. The deviceWISE AEP provides an HTTP interface that enables programmers to create HTTP based applications that either use the TR50 endpoint API or the simple REST API.

Prerequisites

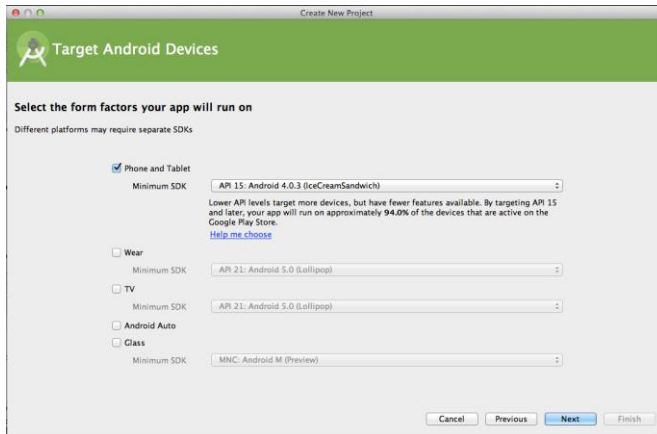
1. [Download](#) and install JDK 7.
2. [Download](#) the Android SDK tools.

Android Project Structure

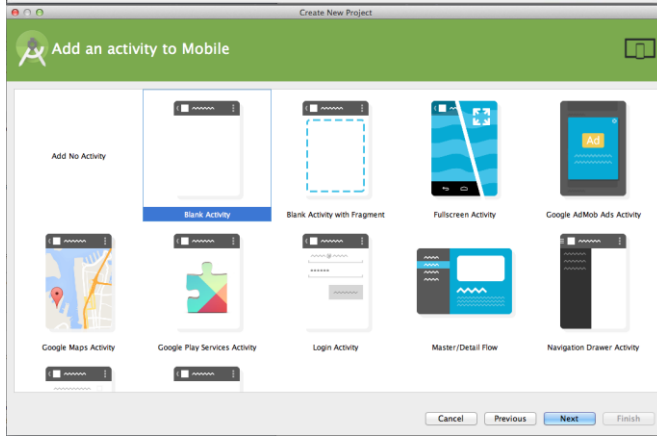
After the necessary plugins have been installed as “Emulator Accelerator” for the development of an Android file development of an Android application can begin. From the top menu of Android Studio, choose File -> New -> New Project.



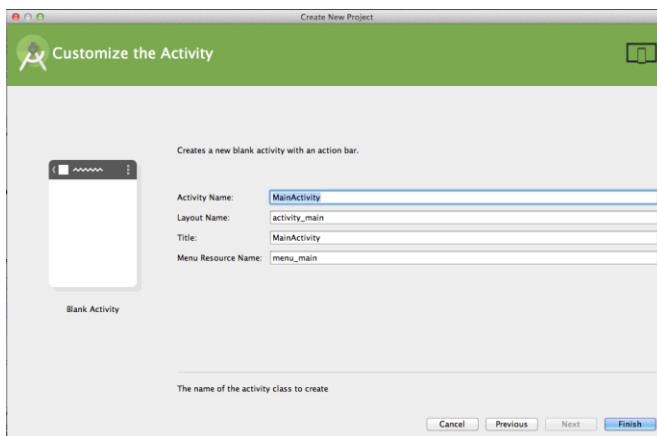
Enter an Application Name and company name of your own. Leave all other fields unchanged and click Next.



Select only Phone and Tablet. In the Minimum SDK Select API 15. Read the description of the minimum SDK to get things clear. Click Next.



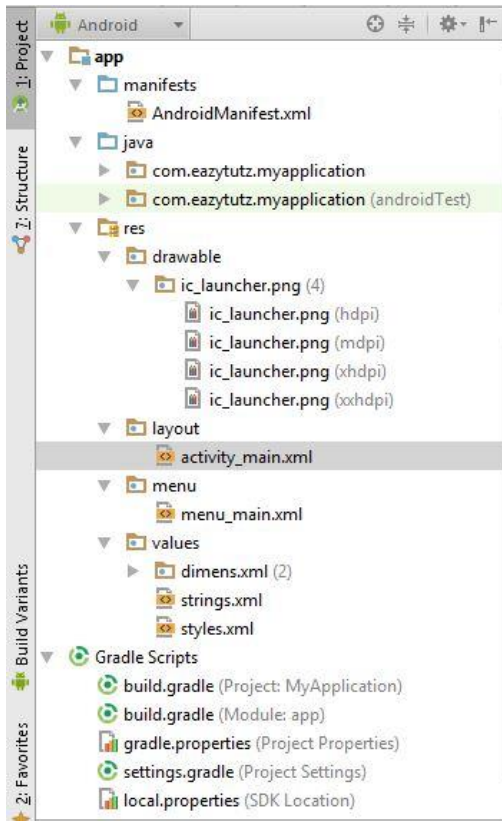
Select Blank Activity and click Next.



Enter an Activity Name of you own. Leave all other fields unchanged and click Finish.

Now the main window of Android Studio opens and in the left most pane of the window, You

can see the Android Project Structure



Here is the brief description of important files/folders in the Android project structure:

1. Manifests

AndroidManifest.xml is one of the most important files in the Android project structure. It contains all the information about the application. When an application is launched, the first file the system seeks is the AndroidManifest.xml file. It actually works as a road map of the application for the system.

The Android Manifest file contains information about:

- Components of the application such as Activities, Services, etc.
- User permissions required
- Minimum level of Android API required

2. Java

The Java folder contains the Java source code files of the application organized into packages. The Android application can have more than one package. It's always a good practice to break the source code of the application into different packages based on its core functionality. All the source files for Activities, Services, etc. go into this folder. In the above screen, the source file of the Activity that was created for our project can be seen.

3. Res

Res folder is where all our external resources for applications are stored, such as images, layout XML files, strings, animations, audio files, etc.

Sub folders:

Drawable

This folder contains the bitmap file to be used in the Application. There are three different folders to store drawable files. They are drawable-ldpi, drawable-mdpi, drawable-hdpi. The folders are to provide alternative image resources for specific screen configurations. Ldpi, mdpi & hdpi stands for low density, medium density & high density screens, respectively. The resources for each screen resolution are stored in respective folders and the android system will choose it according to the pixel density of the device.

Layout

XML files that define the User Interface are placed in this folder.

Values

XML files that define simple values, such as strings, arrays, integers, dimensions, colors, styles, etc. are placed in this folder.

Menu

XML files that define menus in the application are placed in this folder.

4. Gradle Scripts

Gradle scripts are used to automate tasks. This will be covered later. Only understand that Gradle Scripts are used to automate certain tasks. It uses a language called Groovy.

Interface

Layouts:

Main:

Relative layouts are used according to Android best practices. The relative layout works much as its name implies -- it organizes controls relative to one another, or to the parent control itself.

Relative layouts are best explained using an example.

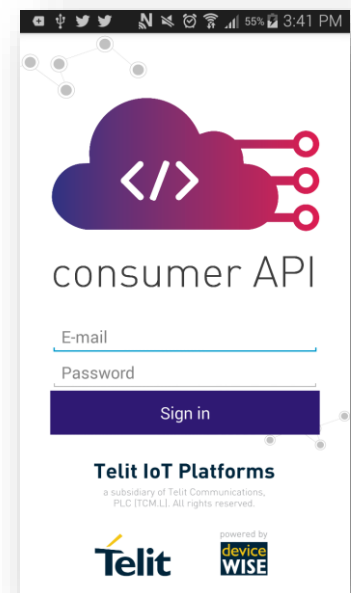
The Login Button has a rule that says:

```
<Button
...
    android:id="@+id/signIn"
    android:layout_marginBottom="161dp"
/>
```

The button will be displayed with a marginBottom value of 161dp, after that every other control will be relative to it. For example:

```
<EditText
...
    android:id="@+id/emailText"
    android:layout_above="@+id/passwordText"
"/>

<EditText
...
    android:id="@+id/passwordText"
    android:layout_above="@+id/signIn"
/>
```

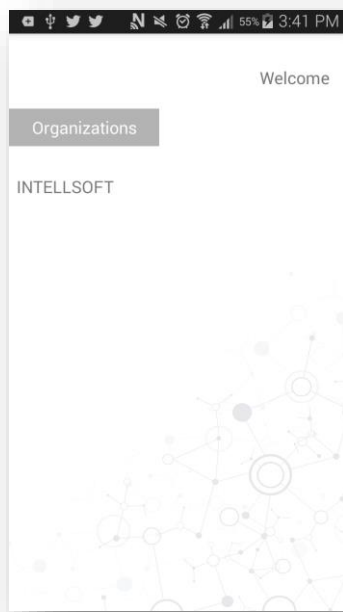


List of Organizations:

Defining an XML Layout Resource with a Relative Layout

The most convenient and maintainable way to design application user interfaces is by creating XML layout resources. This method greatly simplifies the UI design process, moving much of the static creation and layout of user interface controls and definition of control attributes, to the XML, instead of littering the code.

XML layout resources must be stored in the /res/layout project directory hierarchy. Look at the relative layout introduced in the previous section. This layout resource file, aptly named /res/layout/relative.xml, is defined in XML as follows:



<RelativeLayout

```
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    tools:context="com.ilstechnology.androidkitilsv2.organizations"

    android:background="@mipmap/background_net">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome"
        android:id="@+id/welcome_tv"
```

```

android:layout_marginTop="35dp"
android:layout_alignParentTop="true"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true" />

```

...

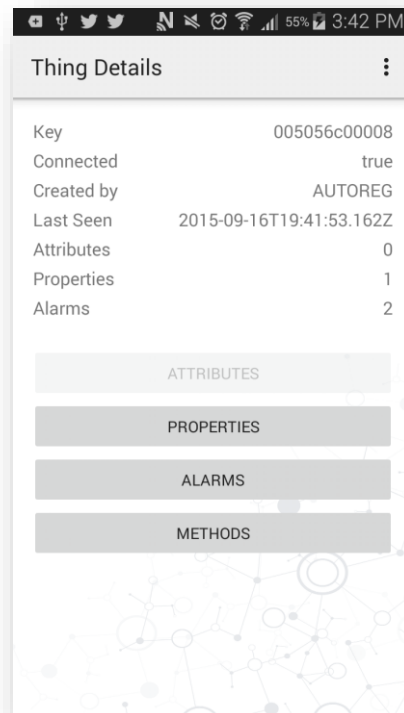
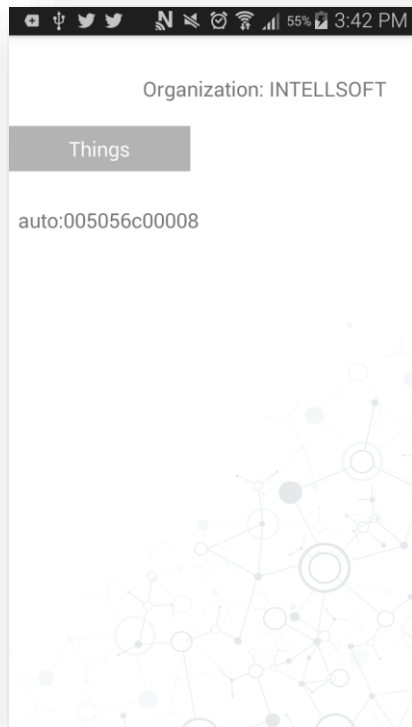
```

<ListView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@id/android:list"
    android:layout_below="@+id/welcome_tv"
    android:layout_marginTop="80dp" />

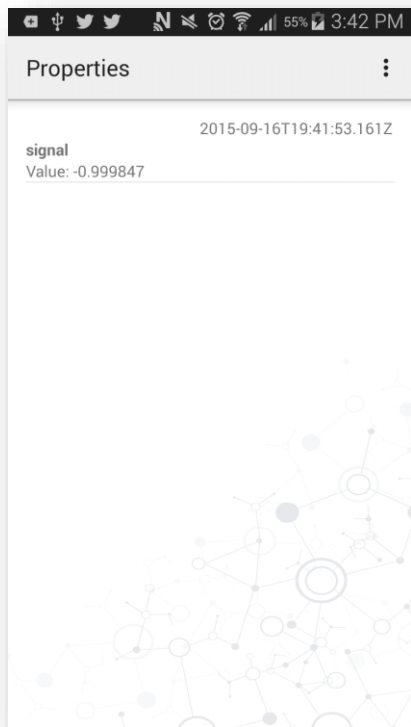
```

```
</RelativeLayout>
```

List of Things & Thing details:



Thing Details:



Backend

Major files in the Android project:

- **AndroidManifest.xml:** This is the Android definition file. It contains information about the Android application such as minimum Android version, permission to access Android device capabilities, such as internet access permission, ability to use phone permission, etc.
- **MainLayout.xml:** This file describes the layout of the page. This means the placement of every component (such as textboxes, labels, radio buttons, user defined components, etc.) on the app screen.
- **Activity class:** Every application that occupies the entire device screen needs at least one class which inherits from the Activity class. One major method is called `onCreate`. This method initiates the app and loads the layout page.

The Android manifest file

A typical Android Manifest file looks like this:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="7" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MynewprojectActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

Some important definitions in this file:

- android:icon: In this path you will find the app's icon
- android:label: App's title
- android:versionCode: This is a numerical value. This is an important attribute because that is how the Android device knows when to alert the user to upgrade the application to a newer version.
- android:sdKMinVersion: Defines what is the earliest version of the Android operating system available to this application.
- Activity element: Defines what the activities available in this application are and which activity has to be loaded at startup.

Layout file

A typical layout file looks like this:

```

<?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" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

</LinearLayout>

```

This layout defines one object, a TextView. This is a text label and it is used when a programmer wants to place a text title in your application.

Activity class

One major class is the Activity class. We already defined this class in the android.manifest file as

the main activity class. Which is the first class to be loaded and executed right after the Android application will be launched:

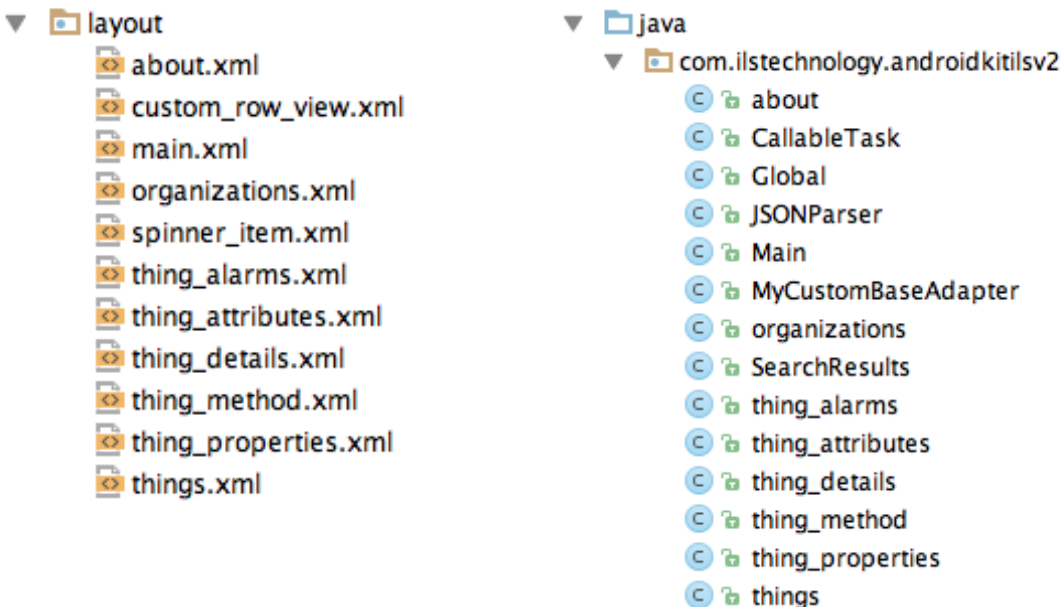
```
package com.firstproject;

import android.app.Activity;
import android.os.Bundle;

public class MynewprojectActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

We can see here one major method called "onCreate" which is called when the Activity class is activated. In our specific example, we define a layout file by the line setContentView(R.layout.main).

Documents structure



Classes

about: Displays succinct information about the application itself, along with high-level details of its creation, a brief summary of the data-feeding platform behind it and our company Telit.

CallableTask: Callable task that returns the list of organizations

Global: global variables.

JSONParser: Use the JSONParser methods to parse a response that's returned from a call to an external service that is in JSON format, such as a JSON-encoded response of a Web service callout.

Main: displays the login screen

Api Calls: api.authenticate, user.find.

MycustomBaseAdapter: A class created to show a table with 3 rows on each row.

Organizations: List of organizations

SearchResults: Class to get and set to the table with multiple elements.

thing_alarms: List of alarms corresponding to an specific thing.

thing_attributes: List of attributes corresponding to an specific thing.

thing_details: Thing's details.

thing_method: List of methods corresponding to an specific thing.

thing_properties: List of properties corresponding to an specific thing.

things: List of things

This app code you will find at the end of the document.

Authentication – “Main” screen

api.authenticate

Authentication Method with login credentials:

//Method: onCreate

```
String json_ = "{\"auth\": {\"command\": \"api.authenticate\", \"params\": {\"password\": \"\"+password_+\"\", \"username\": \"\"+email_+\"\"}}}\";
response_ = jSon.xHttpPost("http://api.devicewise.com/api", json_);
try {
    JSONObject jsonObj = new JSONObject(response_);
    if(jsonObj.has("auth")){
        JSONObject auth = jsonObj.getJSONObject("auth");
        boolean success = auth.getBoolean("success");
        if (success) {
            JSONObject params = auth.getJSONObject("params");
            Global.sessionId = params.getString("sessionId");
            Global.prevOrg = params.getString("orgKey");
            personaldata();
            //Navigate from main to organization screen
            Intent i = new Intent(Main.this, organizations.class);
            startActivity(i);
            //remove the user and password information for security purposes
            ((EditText) findViewById(R.id.emailText)).getText().clear();
            ((EditText) findViewById(R.id.passwordText)).getText().clear();
        }
    }
}
```

```

}else{

    JSONArray error = jsonObj.getJSONArray("errorMessages");
    er = error.get(0).toString();
    //UI Thread to alert the user and password are incorrect
    runOnUiThread(new Runnable() {
        public void run() {
            Toast.makeText(getApplicationContext(), er, Toast.LENGTH_SHORT).show();
        }
    });

}
} catch (Exception e) {
    e.printStackTrace();
}
}

```

API CALL PARAMS:

username: Management Portal user ID (email address)
password: *****

The first TR50 message to be sent from the application to the API must be an authentication command to start a new session and retrieve the **sessionId** parameter. This session ID parameter will be used for any further API communication.

Below is the JSON for a user authentication request and response.

TR50 Request

```

{
  "auth" : {
    "command" : "api.authenticate",
    "params" : {
      "username": "username@example.com",
      "password": "swordfish"
    }
  }
}

```

TR50 Response

```

{
  "auth" : {
    "success" : true,
    "params" : {
      "orgKey": "xxxxx",
      "sessionId": "548716017ed14b29e403d4d0"
    }
  }
}

```

```

    }
  }
}

```

After the authentication is successful, the **sessionId** parameter must be used in any other API calls.

user.find

This method is used to get the name and last name of the user and show them on the Organization screen.

```

//Method: PersonalData
try{
    final JSONParser jSon = new JSONParser();
    String json_ = "{\"auth\":{\"sessionId\":\"" + Global.sessionId +
"\",\"cmd\":{\"command\":\"user.find\", \"params\":{\"emailAddress\":\"" + email.getText() +
"\",\"}}}}";
    response_ = jSon.xHttpPost("http://api.devicewise.com/api", json_);
    JSONObject jsonObj = new JSONObject(response_);
    JSONObject auth = jsonObj.getJSONObject("cmd");
    boolean success = auth.getBoolean("success");

    if (success) {
        JSONObject params = auth.getJSONObject("params");
        String firstName = params.getString("firstName");
        String lastName = params.getString("lastName");
        Global.name = firstName + " " + lastName;
    }

}catch(Exception e){

}

```

API CALL PARAMS:

emailAddress: The email address of the user.

TR50 Request

```
{
  "cmd": {
    "command": "user.find",
    "params": {
      "emailAddress": "user@example.com"
    }
  }
}
```

TR50 Response

```
{
  "cmd": {
    "success": true,
    "params": {
      "id": "52a1d3db169efd03217940d",
      "emailAddress": "user@example.com",
      "defaultOrgId": "fd03217940d52a1d3db169e1",
      "firstName": "Jane",
      "lastName": "Smith",
      "company": "Acme",
      "title": "CEO",
      "officePhone": "5558675309",
      "mobilePhone": "5558675309",
      "isSuperAdmin": false,
      "isSuperOps": false,
      "createdBy": "SYSTEM",
      "createdOn": "2013-12-06T08:40:43.885-05:00",
      "updatedBy": "SYSYEM",
      "updatedOn": "2013-12-06T08:40:43.885-05:00"
    }
  }
}
```

Organizations – “Organizations” screen

Api Call: [Session.org.list](#)

Callable is a thread that returns the result. The `callableTask` was used with the intention to get the organizations and show them in the UI. Otherwise the UI thread doesn't wait for the result and the interface doesn't show the populated list.

Class: CallableTask

```
public String call() throws Exception {
    JSONParser jSon = new JSONParser();
    String json_ = "{ \"auth\": { \"sessionId\": \"\" + Global.sessionId + \"\" }, \"data\":
```

```
{\"command\": \"session.org.list\"}}";
    return jSon.xHttpPost("http://api.devicewise.com/api", jSon_);
}
```

Class: Organization

```
protected void onCreate(Bundle savedInstanceState) {
    final JSONParser jSon = new JSONParser();
    try{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.organizations);
        TextView text = (TextView) findViewById(R.id.welcome_tv);
        if(Global.name == null){
            text.setText("Welcome ");
        }else{
            text.setText("Welcome, " + Global.name + " ");
        }
        ExecutorService executorService = Executors.newFixedThreadPool(1);
        Future<String> future= executorService.submit(new CallableTask());
        response_ = future.get();
        JSONObject jsonObj = new JSONObject(response_);
        JSONObject auth = jsonObj.getJSONObject("data");
        boolean success = auth.getBoolean("success");
        final List<String> orgsString;

        if (success) {
            JSONObject params = auth.getJSONObject("params");
            JSONArray orgJsonArray = params.getJSONArray("result");
            System.out.println(orgJsonArray);
            //Recursive Function
            orgsString = addChecked3(orgJsonArray, 0);

            final ListView myListView = (ListView) findViewById(android.R.id.list);
            myListView.setTextFilterEnabled(true);
            myListView.setAdapter(new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, orgsString));
            myListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
                public void onItemClick(AdapterView<?> myAdapter, View myView, int myItemInt,
                long mylng) {
                    String selectedFromList = (String) (myListView.getItemAtPosition(myItemInt));
                    Global.organizations = orgsString.get(myItemInt).trim();
                    Thread thread = new Thread(new Runnable(){
                        @Override
                        public void run() {
                            try {
                                if (!Global.prevOrg.equals(Global.organizations)){
                                    String json = "{\"auth\":{\"sessionId\":\"\" + Global.sessionId +
                                    \"\"},\"cmd\":{\"command\":\"session.org.switch\", \"params\":{\"key\":\"\" +
                                    Global.organizations + \"\"}}}}";
```

```

        response_ = jSon.xHttpPost("http://api.devicewise.com/api", json);
    }
    Intent i = new Intent(organizations.this, things.class);
    startActivity(i);
} catch (Exception e) {
    e.printStackTrace();
}
}
});
thread.start();

}
});
}

} catch (Exception e) {
    e.printStackTrace();
}
}

```

TR50 Request

```

{
  "cmd": {
    "command": "session.org.list",
    "params": {
      "includeRoles": false
    }
  }
}

```

TR50 Response

```

{
  "cmd": {
    "success": true,
    "result": [
      {
        "orgId": "52fbed495d80f1091b00000f",
        "orgKey": "DEMO",
        "subOrgs": [
          {
            "orgId": "5356af695d80f132b80087a8",
            "orgKey": "GEMS"
          },
          ...
        ]
      }
    ]
  }
}

```

```
}
```

Thing List - “things” screen

Api Call: Thing.list

The thing.list command is used to find and return a list of things.

```
// Method: onCreate function
```

```
//Remove title bar
```

```
this.requestWindowFeature(Window.FEATURE_NO_TITLE);
```

```
setContentView(R.layout.things);
```

```
TextView text = (TextView) findViewById(R.id.org);
```

```
if(Global.organizations == null){
```

```
    text.setText("");
```

```
}else{
```

```
    text.setText("Organization: " + Global.organizations + " ");
```

```
}
```

```
Thread t = new Thread(new Runnable() {
```

```
    public void run() {
```

```
        //Request
```

```
        String json_ = "{\"auth\": {\"sessionId\": \"" + Global.sessionId + "\", \"data\": {\"command\": \"thing.list\", \"params\": {\"limit\": \"200\", \"offset\": 0}}}}";
```

```
        response_ = json.xHttpPost("http://api.devicewise.com/api", json_);
```

```
        try {
```

```
            //Response
```

```
            System.out.println("-----Things-----"+response_);
```

```
            JSONObject jsonObj = new JSONObject(response_);
```

```
            JSONObject auth = jsonObj.getJSONObject("data");
```

```
            boolean success = auth.getBoolean("success");
```

```
            if (success) {
```

```
                JSONObject params = auth.getJSONObject("params");
```

```
                JSONArray mJSONArray = params.getJSONArray("result");
```

```
                int j = 0;
```

```
                JSONObject mJsonObject = new JSONObject();
```

```
                for (int i = 0; i < mJSONArray.length(); i++) {
```

```
                    mJsonObject = mJSONArray.getJSONObject(i);
```

```
                    thingConn.add(mJsonObject.getBoolean("connected"));
```

```
                    thingKey.add(mJsonObject.getString("key"));
```

```
                    thingName.add(mJsonObject.getString("name"));
```

```
                    j++;
```

```
                }
```

```
            }
```

```

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

});
t.start();
try {
    t.join();
} catch (InterruptedException e) {
    e.printStackTrace();
}

Thread h = new Thread(new Runnable() {
    public void run() {
        final ListView myListView = (ListView) findViewById(android.R.id.list);
        //Populating the list
        myListView.setAdapter(new ArrayAdapter<String>(things.this,
        android.R.layout.simple_list_item_1, thingName));
        //Clicking an element of the list
        myListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            public void onItemClick(AdapterView<?> myAdapter, View myView, int myItemInt, long
            myLng) {
                String selectedFromList = (String) (myListView.getItemAtPosition(myItemInt));
                System.out.println("===Thing Position=====" + myItemInt);
                System.out.println("===ID=====" + thingKey.get(myItemInt));
                Global.thing_selected = thingKey.get(myItemInt);
                Intent i = new Intent(things.this, thing_details.class);
                startActivity(i);
            }
        });
    }
});
h.start();

```

TR50 Request

```

{
  "cmd": {
    "command": "thing.list"
  }
}

```

TR50 Response

```

{
  "cmd": {
    "success": true,
    "params": {
      "count": 42,
      "field": [
        ...
      ],
    },
    "result": [
      {
        "id": "531c907b5d80f1330c238757",
        "key": "mything",
        "name": "My Thing",
        "type": "A Thing",
        "connected": true,
        "connectedId": "53404db65d80f12689016ed3",
        "connectedKey": "myconnectedthing",
        "lastSeen": "0001-01-01T00:00:00Z",
        "gateway": {
          "make": "unknown",
          "model": "unknown",
          "dwProduct": "DW-Gateway.Linux",
          "dwPlatform": "Linux-X86-Generic",
          "dwVersion": "14.1.0-014",
          "appVersion": "2.0",
          "remShell": true
        },
        "proto": "mqtt",
        "remoteAddr": "127.0.0.1:60709",
        "locWithin": {
          "": ""
        },
        "locUpdated": "2014-03-27T13:39:26.635-04:00",
        "loc": {
          "lat": 30.455,
          "lng": -84.253333,
          "acc": 0,
          "fixType": "unknown",
          "addr": {
            "streetNumber": "",
            "street": "1113 Buckingham Dr",
            "city": "Tallahassee",
            "state": "FL",
            "zipCode": "32308-5212",
            "country": "US"
          }
        }
      }
    ],
  },
}

```

```
...  
]  
}  
}  
}
```

Thing Details – “thing_details” screen

API Call: Thing.find

The thing.find command is used to find and return a thing. With this API properties, attributes and alarms and other items can be retrieved, such as data point like thing created by, last seen (if the thing is connected).

```
//Method: onCreate
Thread thread = new Thread(new Runnable(){
    @Override
    public void run() {
        try {
//Request
            String json = "{\"auth\":{\"sessionId\":\"\" + Global.sessionId +
            \"\"},\"cmd\":{\"command\":\"thing.find\",\"params\":{\"key\":\"\" + Global.thing_selected +
            \"\"}}}\";
            String response_ = jSon.xHttpPost("http://api.devicewise.com/api", json);
//Response
            JSONObject jsonObj = new JSONObject(response_);
            JSONObject auth = jsonObj.getJSONObject("cmd");
            boolean success = auth.getBoolean("success");
            if (success) {
                JSONObject params = auth.getJSONObject("params");
                propName.clear();
                propValue.clear();
                propTs.clear();
                if(params.has("properties")) {
                    //props
                    JSONObject jsonProps = params.getJSONObject("properties");
                    Iterator<?> keys = jsonProps.keys();
                    while( keys.hasNext() ) {
                        String key = (String)keys.next();
                        JSONObject value = jsonProps.getJSONObject(key);
                        String val = value.getString("value");
                        String ts = value.getString("ts");
                        propName.add(key);
                        propValue.add("Value: " + val);
                        propTs.add(ts);
                        countProp++;
                    }
                }
                else{
                    countProp = 0;
                }
                attrsName.clear();
                attrsValue.clear();
                attrsTs.clear();
                if(params.has("attrs")) {
```



```

        JSONObject jsonAttrs = params.getJSONObject("attrs");
        Iterator<?> keys = jsonAttrs.keys();
        while( keys.hasNext() ) {
            String key = (String)keys.next();
            JSONObject value = jsonAttrs.getJSONObject(key);
            String val = value.getString("value");
            String ts = value.getString("ts");
            attrsName.add(key);
            attrsValue.add("Value: " + val);
            attrsTs.add(ts);
            countAttrs++;
        }
    }else{
        countAttrs = 0;
    }
    alarmState.clear();
    alarmName.clear();
    alarmTs.clear();
    if(params.has("alarms")) {
        JSONObject jsonAlarms = params.getJSONObject("alarms");
        Iterator<?> keys = jsonAlarms.keys();
        while( keys.hasNext() ) {
            String key = (String)keys.next();
            JSONObject value = jsonAlarms.getJSONObject(key);
            String val = value.getString("state");
            String ts = value.getString("ts");
            alarmName.add(key);
            alarmState.add("State: " + val);
            alarmTs.add(ts);
            countAlarms++;
        }
    }else{
        countAlarms = 0;
    }
    createdby = params.getString("createdBy");
    connected = params.getBoolean("connected");
    lastSeen = params.getString("lastSeen");
    defKey = params.getString("defKey");
}
// }

} catch (Exception e) {
    e.printStackTrace();
}
}
});
thread.start();
try {

```

```

    thread.join();
} catch (InterruptedException e) {
    e.printStackTrace();
}
... // Check the rest of the code in the project
});

```

API CALL PARAMS:

key : Identifies the thing.

TR50 Request

```

{
  "cmd": {
    "command": "thing.find",
    "params": {
      "key": "mything"
    }
  }
}

```

TR50 Response

```

{
  "cmd": {
    "success": true,
    "params": {
      "id": "52f1413a63c4b80b5700007c",
      "name": "My Thing",
      "key": "mything",
      "desc": "Lorem ipsum dolor sit amet...",
      "defId": "52f105e363c4b87f38000700",
      "defKey": "default",
      "secTags": [
        "sectag-a",
        "sectag-b"
      ],
      "tags": [
        "tag1",
        "tag2"
      ],
      "lastSeen": "0001-01-01T00:00:00Z",
      "locEnabled": true,
      "locUpdated": "2014-03-27T13:39:26.635-04:00",
      "loc": {
        "lat": 30.455,
        "lng": -84.253333,
        "fixType": "unknown",
        "addr": {

```

```

        "streetNumber": "",
        "street": "1113 Buckingham Dr",
        "city": "Tallahassee",
        "state": "FL",
        "zipCode": "32308-5212",
        "country": "US"
    }
},
"tunnelActualHost": "192.168.4.241",
"tunnelVirtualHost": "127.10.10.10",
"tunnelLatencies": {
    "router01": 110,
    "router02": 340
},
"createdBy": "admin@devicewise.com",
"createdOn": "2014-02-04T14:36:26.783-05:00",
"updatedBy": "admin@devicewise.com",
"updatedOn": "2014-03-27T11:06:32.758-04:00",
"connected": false
}
}
}

```

Alarms – “thing_alarms” screen

This function populates the table with alarm Name, alarm Status and alarm TimeStamp. We add the function of Refresh. The user can swipe the screen vertically and the values will be refreshed.

Java Method: onCreate()

```

super.onCreate(savedInstanceState);
setContentView(R.layout.thing_alarms);

```

```

final SwipeRefreshLayout swipeView = (SwipeRefreshLayout) findViewById(R.id.swipeAlarms);
swipeView.setColorScheme(android.R.color.holo_blue_dark, android.R.color.holo_blue_light,
    android.R.color.holo_green_light, android.R.color.holo_orange_light);
swipeView.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
    @Override
    //Refresh Function
    public void onRefresh() {
        swipeView.setRefreshing(true);
        (new Handler()).postDelayed(new Runnable() {
            @Override
            public void run() {
                swipeView.setRefreshing(false);
                apiProps();
                auxFunction();
            }
        }, 1000);
    }
});

```

```

    }, 3000);
}
});

```

auxFunction();

Java Method: auxFunction()

```

//Populate the list with multielements
private void auxFunction(){
    ArrayList<SearchResults> searchResults = GetSearchResults();

    final ListView lv1 = (ListView) findViewById(R.id.List_alarms);
    lv1.invalidateViews();
    lv1.setAdapter(new MyCustomBaseAdapter(this, searchResults));

    lv1.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> a, View v, int position, long id) {
            Object o = lv1.getItemAtPosition(position);
            SearchResults fullObject = (SearchResults) o;
            Toast.makeText(thing_alarms.this, "You have chosen: " + " " + fullObject.getName(),
            Toast.LENGTH_LONG).show();
        }
    });
}

```

Java Method: apiProps()

```

//This function is called only after the user refresh the get new values.
//This api was already explained.
private void apiProps(){
    Thread thread = new Thread(new Runnable(){
        @Override
        public void run() {
            try {
                String json = "{\"auth\":{\"sessionId\":\"\" + Global.sessionId +
                \"\"},\"cmd\":{\"command\":\"thing.find\",\"params\":{\"key\":\"\" + Global.thing_selected +
                \"\"}}}\";

                String response_ = jSon.xHttpPost("http://api.devicewise.com/api", json);
                System.out.println(json + "-----Response Thing-----" + response_);

                JSONObject jsonObj = new JSONObject(response_);
                JSONObject auth = jsonObj.getJSONObject("cmd");
                boolean success = auth.getBoolean("success");
                if (success) {

```

```

JSONObject params = auth.getJSONObject("params");

thing_details.alarmState.clear();
thing_details.alarmName.clear();
thing_details.alarmTs.clear();
if(params.has("alarms")) {
    JSONObject jsonAlarms = params.getJSONObject("alarms");
    Iterator<?> keys = jsonAlarms.keys();
    while( keys.hasNext() ) {
        String key = (String)keys.next();
        JSONObject value = jsonAlarms.getJSONObject(key);
        String val = value.getString("state");
        String ts = value.getString("ts");
        thing_details.alarmName.add(key);
        thing_details.alarmState.add("State: " + val);
        thing_details.alarmTs.add(ts);
        countAlarms++;
    }
} else {
    countAlarms = 0;
}
} catch (Exception e) {
    e.printStackTrace();
}
}
});
thread.start();
try {
    thread.join();
} catch (InterruptedException e) {
    e.printStackTrace();
}
}
}

```

Java Method: GetSearchResults()

```

//Set the array with the values of the alarms corresponding to a specific thing.
private ArrayList<SearchResults> GetSearchResults(){
    ArrayList<SearchResults> results = new ArrayList<SearchResults>();
    SearchResults sr1 = new SearchResults();

    for (int i=0; i < thing_details.alarmName.size(); i++) {

        sr1.setName(thing_details.alarmName.get(i));
        sr1.setValue(thing_details.alarmState.get(i));
        sr1.setTs(thing_details.alarmTs.get(i));
    }
}

```

```

        results.add(sr1);
    }
    return results;
}

```

Attributes – “thing_attributes” screen

This function populates the table with attribute Name, attribute Value and attribute TimeStamp. A function to Refresh is in this application. The user can swipe the screen vertically and the values will be refreshed.

Java Method: onCreate()

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.thing_attributes);

    final SwipeRefreshLayout swipeView = (SwipeRefreshLayout)
    findViewById(R.id.swipeAttributes);
    swipeView.setColorScheme(android.R.color.holo_blue_dark, android.R.color.holo_blue_light,
    android.R.color.holo_green_light, android.R.color.holo_orange_light);
    swipeView.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
        @Override
        public void onRefresh() {
            swipeView.setRefreshing(true);
            (new Handler()).postDelayed(new Runnable() {
                @Override
                public void run() {
                    swipeView.setRefreshing(false);
                    apiAttrs();
                    auxFunction();
                }
            }, 3000);
        }
    });

    auxFunction();
}

```

Java Method: auxFunction()

```

//Populate the list with multielements
private void auxFunction(){
    ArrayList<SearchResults> searchResults = GetSearchResults();

    final ListView lv1 = (ListView) findViewById(R.id.List_attr);
}

```

```

lv1.invalidateViews();
lv1.setAdapter(new MyCustomBaseAdapter(this, searchResults));

lv1.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> a, View v, int position, long id) {
        Object o = lv1.getItemAtPosition(position);
        SearchResults fullObject = (SearchResults) o;
        Toast.makeText(thing_attributes.this, "You have chosen: " + " " + fullObject.getName(),
Toast.LENGTH_LONG).show();
    }
});
}

```

Java Method: apiAttrs()

//This function is called only after the user refresh the get new values.

//This api was already explained.

```

private void apiAttrs(){

    Thread thread = new Thread(new Runnable(){
        @Override
        public void run() {
            try {
                String json = "{\"auth\":{\"sessionId\":\"\" + Global.sessionId +
                \"\"},\"cmd\":{\"command\":\"thing.find\",\"params\":{\"key\":\"\" + Global.thing_selected +
                \"\"}}}\"";
                String response_ = jSon.xHttpPost("http://api.devicewise.com/api", json);

                JSONObject jsonObj = new JSONObject(response_);
                JSONObject auth = jsonObj.getJSONObject("cmd");
                boolean success = auth.getBoolean("success");
                if (success) {
                    JSONObject params = auth.getJSONObject("params");
                    thing_details.attrsName.clear();
                    thing_details.attrsValue.clear();
                    thing_details.attrsTs.clear();
                    if(params.has("attrs")) {
                        JSONObject jsonAttrs = params.getJSONObject("attrs");
                        Iterator<?> keys = jsonAttrs.keys();
                        while( keys.hasNext() ) {
                            String key = (String)keys.next();
                            JSONObject value = jsonAttrs.getJSONObject(key);
                            String val = value.getString("value");
                            String ts = value.getString("ts");
                            thing_details.attrsName.add(key);
                            thing_details.attrsValue.add("Value: " + val);
                        }
                    }
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

```

```

        thing_details.attrsTs.add(ts);
        countAttrs++;
    }
    }else{
        countAttrs = 0;
    }
    }
    } catch (Exception e) {
        e.printStackTrace();
    }
    }
    });
    thread.start();
    try {
        thread.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

```

Java Method: GetSearchResults()

```

//Set the array with the values of the alarms corresponding to a specific thing.
private ArrayList<SearchResults> GetSearchResults(){
    ArrayList<SearchResults> results = new ArrayList<SearchResults>();
    SearchResults sr1 = new SearchResults();
    ArrayList<String> hola = (ArrayList<String>) thing_details.attrsName;
    for (int i=0; i < thing_details.attrsName.size(); i++) {
        sr1.setName(thing_details.attrsName.get(i));
        sr1.setValue(thing_details.attrsValue.get(i));
        sr1.setTs(thing_details.attrsTs.get(i));
        results.add(sr1);
    }
    return results;
}

```

Properties – “thing_properties” screen

This function populates the table with property Name, property Value and property TimeStamp. A function to Refresh is in this application. The user can swipe the screen vertically and the values will be refreshed.

Java Method: onCreate()

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.thing_properties);
}

```



```

        final SwipeRefreshLayout swipeView = (SwipeRefreshLayout)
findViewById(R.id.swipeProperties);
        swipeView.setColorScheme(android.R.color.holo_blue_dark, android.R.color.holo_blue_light,
android.R.color.holo_green_light, android.R.color.holo_orange_light);
        swipeView.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {
                swipeView.setRefreshing(true);
                // Log.d("Swipe", "Refreshing Number");
                (new Handler()).postDelayed(new Runnable() {
                    @Override
                    public void run() {
                        swipeView.setRefreshing(false);
                        apiProps();
                        auxFunction();
                    }
                }, 3000);
            }
        });

        auxFunction();
    }

```

Java Method: auxFunction()

```

//Populate the list with multielements
private void auxFunction(){
    ArrayList<SearchResults> searchResults = GetSearchResults();

    final ListView lv1 = (ListView) findViewById(R.id.List_properties);
    lv1.invalidateViews();
    lv1.setAdapter(new MyCustomBaseAdapter(this, searchResults));

    lv1.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> a, View v, int position, long id) {
            Object o = lv1.getItemAtPosition(position);
            SearchResults fullObject = (SearchResults) o;
            Toast.makeText(thing_attributes.this, "You have chosen: " + " " + fullObject.getName(),
Toast.LENGTH_LONG).show();
        }
    });
}

```

Java Method: apiAttrs()

//This function is called only after the user refresh the get new values.

//This api was already explained.

```
private void apiProps(){

    Thread thread = new Thread(new Runnable(){
        @Override
        public void run() {
            try {
                // if (!Global.prevOrg.equals(Global.organizations)){
                String json = "{\"auth\":{\"sessionId\":\"\" + Global.sessionId +
                \"\"},\"cmd\":{\"command\":\"thing.find\",\"params\":{\"key\":\"\" + Global.thing_selected +
                \"\"}}}\";

                String response_ = jSon.xHttpPost("http://api.devicewise.com/api", json);
                System.out.println(json + "-----Response Thing-----" + response_);

                JSONObject jsonObj = new JSONObject(response_);
                JSONObject auth = jsonObj.getJSONObject("cmd");
                boolean success = auth.getBoolean("success");
                if (success) {
                    JSONObject params = auth.getJSONObject("params");

                    thing_details.propName.clear();
                    thing_details.propValue.clear();
                    thing_details.propTs.clear();
                    if(params.has("properties")) {
                        //props
                        JSONObject jsonProps = params.getJSONObject("properties");
                        Iterator<?> keys = jsonProps.keys();
                        while( keys.hasNext() ) {
                            String key = (String)keys.next();
                            JSONObject value = jsonProps.getJSONObject(key);
                            String val = value.getString("value");
                            String ts = value.getString("ts");
                            thing_details.propName.add(key);
                            thing_details.propValue.add("Value: " + val);
                            thing_details.propTs.add(ts);
                            countProp++;
                        }
                    }else{
                        countProp = 0;
                    }
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}
```

```

    }
});
thread.start();
try {
    thread.join();
} catch (InterruptedException e) {
    e.printStackTrace();
}
}
}

```

Java Method: GetSearchResults()

```

//Set the array with the values of the alarms corresponding to a specific thing.
private ArrayList<SearchResults> GetSearchResults(){
    ArrayList<SearchResults> results = new ArrayList<SearchResults>();
    SearchResults sr1;

    for (int i=0; i < thing_details.propName.size(); i++) {
        sr1 = new SearchResults();
        sr1.setName(thing_details.propName.get(i));
        sr1.setValue(thing_details.propValue.get(i));
        sr1.setTs(thing_details.propTs.get(i));
        results.add(sr1);
    }
    return results;
}

```

Method – “thing_method” screen

This function shows the method name and the list of attributes for the method.

Java Class: MethodDetail

```

//Every method has attributes with a value and type corresponding to that value. In this class
with are getting array with that information.
public class detailMethod {
    ArrayList<String> value;
    ArrayList<String> type;

    //constructor
    public detailMethod(){
        value = new ArrayList<String>();
        type = new ArrayList<String>();
    };

    public void add(String a, String b) {
        value.add(a);
    }
}

```

```

        type.add(b);
    }
}

```

Java Class: method

```

//Class with the method name and MethodDetail
public class method {
    String name;
    detailMethod arrayMethods;
    //constructor
    method(String n, detailMethod d) {
        this.name = n;
        this.arrayMethods = d;
    }

    method(){}

    String getName(){
        return name;
    }
}

```

API Call: thing_def.find

The thing_def.find command is used to find an existing Thing Definition.

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.thing_method);
    final Thread thread = new Thread(new Runnable(){

        public void run() {
            try {

                String json = "{\"auth\":{\"sessionId\":\"\" + Global.sessionId +
                \"\"},\"cmd\":{\"command\":\"thing_def.find\",\"params\":{\"key\":\"\" + thing_details.defKey +
                \"\"}}}\";

                String response_ = jSon.xHttpPost("http://api.devicewise.com/api", json);
                System.out.println(json + "-----Response Thing Definition-----" + response_);

                JSONObject jsonObj = new JSONObject(response_);
                JSONObject auth = jsonObj.getJSONObject("cmd");
                boolean success = auth.getBoolean("success");
                if (success) {
                    JSONObject params = auth.getJSONObject("params");

```

```

        if(params.has("methods")) {
            //props
            JSONObject jsonMethods = params.getJSONObject("methods");
            Iterator<?> keys = jsonMethods.keys();
            while( keys.hasNext() ) {

                String key = (String)keys.next();
                detailMethod det = new detailMethod();

                JSONObject value = jsonMethods.getJSONObject(key);
                JSONObject notificationVar = value.getJSONObject("notificationVariables");
                Iterator<?> sub_keys = notificationVar.keys();
                while( sub_keys.hasNext() ) {
                    String sub_key = (String)sub_keys.next();
                    JSONObject value_ = jsonMethods.getJSONObject(key);
                    JSONObject valueByMethod_ = notificationVar.getJSONObject(sub_key);
                    String getValue = valueByMethod_.getString("type");
                    det.add(sub_key, getValue );
                }

                meth.add(new method(key, det));
                nameList.add(key);

            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

});
thread.start();
try {
    thread.join();
} catch (InterruptedException e) {
    e.printStackTrace();
}
}

runOnUiThread(new Runnable() {
    @Override
    public void run() {
        //Methods dropdown
        Spinner dropdown = (Spinner) findViewById(R.id.method);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(getApplicationContext(),
R.layout.spinner_item, nameList);
        dropdown.setAdapter(adapter);
        dropdown.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View selectedItemView, int pos,

```

```

long id) {
    Toast.makeText(getApplicationContext(), "The method selected is " +
        parent.getItemAtPosition(pos).toString(), Toast.LENGTH_LONG).show();
    methodSelected = parent.getItemAtPosition(pos).toString();

    for (int i = 0; i < meth.size(); i++) {
        if (parent.getItemAtPosition(pos).toString() == nameList.get(i)) {
            int positionM = i;
            typeList = meth.get(positionM).arrayMethods.type;
            valueList = meth.get(positionM).arrayMethods.value;
        }
    }
    LinearLayout layout = (LinearLayout) findViewById(R.id.ly);
    LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
        android.widget.LinearLayout.LayoutParams.MATCH_PARENT,
        android.widget.LinearLayout.LayoutParams.WRAP_CONTENT);
    layout.setPadding(0, 15, 0, 0);

    int childcount = layout.getChildCount();
    for (int i=0; i < childcount; i++){
        View v = layout.getChildAt(i);
        if(layout.getChildAt(i) instanceof TextView){
            layout.removeView(layout.getChildAt(i));
        }
        if(layout.getChildAt(i) instanceof EditText){
            layout.removeView(layout.getChildAt(i));
        }
        if(layout.getChildAt(i) instanceof Space){
            layout.removeView(layout.getChildAt(i));
        }
        if(layout.getChildAt(i) instanceof Button){
            layout.removeView(layout.getChildAt(i));
        }
    }

    for (int i = 0; i < typeList.size(); i++) {

        // text view
        TextView tv = new TextView(getApplicationContext());
        tv.setLayoutParams(new
TableLayout.LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
        tv.setText(valueList.get(i));
        tv.setTextSize(TypedValue.COMPLEX_UNIT_SP, 18);
        tv.setTextColor(Color.parseColor("#737374"));
        tv.setLayoutParams(params);
        layout.addView(tv);

        EditText edttext = new EditText(getApplicationContext());

```

```

        edtttext.setLayoutParams(new
TableLayout.LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
        edtttext.setLayoutParams(params);
        edtttext.setId(i);
        edtttext.setTextColor(Color.parseColor("#737374"));
        layout.addView(edtttext);
        etList.add(edtttext);
    }

    Space sp = new Space(getApplicationContext());
    LinearLayout.LayoutParams params_ = new LinearLayout.LayoutParams(
        LayoutParams.WRAP_CONTENT, 30);
    sp.setLayoutParams(params_);
    layout.addView(sp);

    Button bt = new Button(getApplicationContext());
    bt.setText("Exec Method");
    bt.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT,
LayoutParams.WRAP_CONTENT));
    layout.addView(bt);
    bt.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Thread thread_ = new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        for (EditText et : etList) {
                            if(et.getText().toString().length() == 0){
                                runOnUiThread(new Runnable() {
                                    public void run() {
                                        Toast.makeText(getApplicationContext(), "Please fill the fields",
Toast.LENGTH_SHORT).show();
                                        getout = true;
                                    }
                                });
                            }
                        }
                        if(getout == true) return;
                        getValueMethod.add(et.getText().toString());
                    }
                    if(getout == true) return;
                    String jsonParams = ", \"params\": {";
                    if (valueList.size() != 0) {
                        for (int i = 0; i < typeList.size(); i++) {

                            if(typeList.get(i).equals("float") || typeList.get(i).equals("int")){
                                jsonParams+= "\"" +valueList.get(i)+"\": "+getValueMethod.get(i);
                            }else{

```

```

        jsonParams+= "\"" +valueList.get(i)+"\":
\""+getValueMethod.get(i)+"\"";
    }
    if (i != typeList.size() - 1 || typeList.size()!= 1) {
        jsonParams+= ",";
    }
    }
    jsonParams+= "}}";
} else {
    jsonParams = "";
}

String json = "{\"auth\": {\"sessionId\": \"\" + Global.sessionId +
\"\", \"cmd\": {\"command\": \"method.exec\", \"params\": {\"ackTimeout\": 10, \"method\": \"\"
+ methodSelected + \"\", \"singleton\": false, \"thingKey\": \"\" + Global.thing_selected + \"\" +
jsonParams + \"}}}";

response_ = jSon.xHttpPost("http://api.devicewise.com/api", json);
System.out.println(json + "-----Response Thing Definition-----" +
response_);

JSONObject jsonObj = new JSONObject(response_);
textArea();

} catch (Exception e) {
    e.printStackTrace();
}
});

thread_.start();
}
});
}
});
}
}

```

API CALL PARAMS:

key : The key of the Thing Definition to find

TR50 Request

```

{
  "cmd": {
    "command": "thing_def.find",
    "params": {
      "key": "mythingdef"
    }
  }
}

```



```

    }
  }
}

```

TR50 Response

```

{
  "cmd": {
    "success": true,
    "params": {
      "id": "531c8f995d80f1330c21acb9",
      "key": "connected_car",
      "name": "Connected Car",
      "version": 31,
      "autoDefProps": false,
      "autoDefAttrs": false,
      "properties": {
        "fuelecon": {
          "name": "Fuel Economy",
          "unit": "MPG"
        },
        ...
      },
      "alarms": {
        ...
      },
      "methods": {
        "setreportinginterval": {
          "name": "Set Reporting Interval",
          "notificationVariables": {
            "engineoff": {
              "name": "Reporting while engine off",
              "type": "int",
              "uiType": "text"
            },
            "engineon": {
              "name": "Reporting while engine on",
              "type": "int",
              "uiType": "text"
            }
          }
        },
        ...
      }
    }
  }
}

```

Java Method: `textArea()`

```
//Method that populate the API response
public void textArea(){
    try {

        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                LinearLayout layout = (LinearLayout) findViewById(R.id.ly);

                Space sp = new Space(getApplicationContext());
                LinearLayout.LayoutParams params_ = new
                LinearLayout.LayoutParams(LayoutParams.WRAP_CONTENT, 30);
                sp.setLayoutParams(params_);
                layout.addView(sp);

                // text view
                TextView tv1 = new TextView(getApplicationContext());
                tv1.setLayoutParams(new
                TableLayout.LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
                tv1.setText("");
                tv1.setTextSize(TypedValue.COMPLEX_UNIT_SP, 18);
                tv1.setTextColor(Color.parseColor("#737374"));
                tv1.setText(response_);
                layout.addView(tv1);

            }
        });
    } catch (Exception e) {
        e.printStackTrace();
    }
}
};
}
```

Processing the Commands

Java Method: `apiCall()`

`apiCall(json)`:this function process the Json Data by POST.

Description:

//URL (endpoint): <https://open-api.devicewise.com/api>, in all-lowercase letters.

```

public String xHttpPost(String targetURL, String jSon){
    URL url;
    HttpURLConnection connection = null;
    try {
        //Create connection
        url = new URL(targetURL);
        //Type: Post

        connection = (HttpURLConnection)url.openConnection();
        connection.setRequestMethod("POST");
        connection.setRequestProperty("Content-Type",
            "application/x-www-form-urlencoded");

        connection.setRequestProperty("Content-Length", "" +
            Integer.toString(jSon.getBytes().length));
        connection.setRequestProperty("Content-Language", "en-US");

        connection.setUseCaches (false);
        connection.setDoInput(true);
        connection.setDoOutput(true);

        //Send request
        DataOutputStream wr = new DataOutputStream (
            connection.getOutputStream ());
        wr.writeBytes (jSon);
        wr.flush();
        wr.close();

        //Get Response
        InputStream is = connection.getInputStream();
        BufferedReader rd = new BufferedReader(new InputStreamReader(is));
        String line;
        StringBuffer response = new StringBuffer();
        while((line = rd.readLine()) != null) {
            response.append(line);
            response.append('\r');
        }
        rd.close();
        System.out.println(response.toString());
        return response.toString();

    } catch (Exception e) {

        e.printStackTrace();
        return "error";

    } finally {

```

```

        if(connection != null) {
            connection.disconnect();
        }
    }
}

```

Success: A callback function that is executed if the request succeeds.

If the answer is successful the request is:

```

{
  "cmd": {
    "success": true
  }
}

```

If not the request is:

```

{
  "cmd": {
    "success": false,
    "errormessages": ["Session 'XXXXX' not found."], "errorcodes": [-90008]
  }
}

```

Mobile project

Double click to this project:



androidKitILSv2.zip

APK:



app-debug.apk