Intercepting Menu Taps / Accessing JSON Values

This purpose of this document is to help you understand how a JSON Screen Object is constructed so you can perform different actions based on different values in the JSON. The easiest way to illustrate how this works is to intercept a simple menu list "tap" event, analyze the JSON, then make process some logic.

In the example, we will be looking at a menu item tapped then opening a default Android Activity if we find that the menu item should open a BT_screen_pdfDoc type screen. Understanding how this works could help you in other situations in other parts of your program.

The setup

We will be working with a simple (relatively) plugin, BT_screen_menuList. This screen uses a built in Android ListView to display a list of items. Every item in the list is "connected" to a screen object that should be loaded when the row is tapped.

Open up the BT_screen_menuList.java file. Scroll down to line 386 and find this line:

```
final OnItemClickListener listItemClickHandler.....
```

This "click listener" is the section of code that fires when any item in the list is tapped. When an item is tapped, the program processes some logic to determine which screen to load next. This "new screen" is the screen we will be analyzing. In this case, this new screen objects is named: tapScreenLoadObject

The code

Scroll down to line 432 where you see this line:

```
if(tapScreenLoadObject != null).....
```

This check is telling the code to continue processing if it was successful in determining what screen needs to load next. In our case, we need to inspect this screen to see if it's a PDF. If it is a PDF, we'll open the PDF in a default activity and NOT load the BT_screen_pdfDoc plugin.

In order to do this, we need to insert some code just after line 432 (like the line above). This code will use a built in buzztouch method (from the BT_strings.java class) to get the details of the new screen we are loading. If these details tell us that it's a PDF type screen, we'll load the PDF. If it's not a PDF type screen, our newly created code will be ignored.

Copy and paste the code on the last page of this document. Paste it just below line 432. This means your newly added code will be in between the code above, and this line...

```
BT_act_controller.loadScreenObject(thisActivity, screenData.....
```

The code you paste is well commented so you should spend some time trying to understand how it works.

Side Note: The `BT_strings.`*`getJsonPropertyValue`* method is especially useful and is used in lots of places throughout your program, including in the new code below. This method takes three arguments and returns one value. When it's called, it's passed a JSON string of data, a property name, and a default value. If the string of JSON data you pass it contains the property name it will return it's value. If the JSON does not contain the property, it will return the default you provide.

When you copy / paste the code on the last page of this document into the BT_screen_menuList.java file, copy eveything, including this starting and ending /////// lines. Including these markers helps make the code “sections” more readable.

```
////////////////////////////////////////////////////////////
/* CHECK TO SEE IF THIS IS A PDF DOC

The tapScreenLoadObject is the JSON data holding the information about the screen
we are about to load. This screen object is easier to conceptualize when you understand how
it looks. Here's a typcial screen object. The JSON values depend on the screen type...

{"itemId":"99999",
        "itemType":"BT_screen_pdfDoc",
        "itemNickname":"PDF Doc",
        "navBarTitleText":"Cool PDF File",
        "dataURL":"http://www.buzztouch.com/resources/bt-server-installation.pdf"
}

We need to check three different values in the JSON. itemType, localFileName and dataURL.
If the itemType = BT_screen_pdfDoc then we know we want to open a PDF doc. We also need to
check for a localFileName or a dataURL to see where the PDF doc is. If it's local (included
in the project) the logic is different than if it's online at a URL. We use a built in buzztouch method
in the BT_strings class to get these values. This method is used in lots and lots of places in your program.
The method takes three arguments. 1) The JSON data (like the example data above). 2) The name of the property we are
looking for. And 3) The default value we want to return if the property does not exist. In this case, we are not
providing a default value (empty string) because we simply want to know if the value exists.
*/

String itemType = BT_strings.getJsonPropertyValue(tapScreenLoadObject.getJsonObject(), "itemType", "");
String dataURL = BT_strings.getJsonPropertyValue(tapScreenLoadObject.getJsonObject(), "dataURL", "");
String localFileName = BT_strings.getJsonPropertyValue(tapScreenLoadObject.getJsonObject(), "localFileName", "");

//if this is a PDF screen, process our new logic...
if(itemType.equalsIgnoreCase("BT_screen_pdfDoc")){
        BT_debugger.showIt(activityName + ":loading PDF doc from menu tap...");

        //if we have a localFileName, make sure it exists before trying to open it...
        if(localFileName.length() > 1){
                if(BT_fileManager.doesProjectAssetExist("BT_Docs", localFileName)){

                        //copy from the BT_Docs folder to the app's cache so we can open it...
                        BT_fileManager.copyAssetToCache("BT_Docs", localFileName);

                        //find the path to the local file in the cache...
                        try{

                                PackageManager pm = getPackageManager();
                                ApplicationInfo appInfo = pm.getApplicationInfo(getPackageName(), 0);
                                File theFile = new File(appInfo.dataDir + "/files/" + localFileName);

                                Intent intent = new Intent();
                                intent.setAction(Intent.ACTION_VIEW);
                                Uri uri = Uri.fromFile(theFile);
                                intent.setDataAndType(uri, "application/pdf");
                                intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                startActivity(intent);

                        }catch(Exception e){
                                BT_debugger.showIt(activityName + ":EXCEPTION " + e.toString());
                        }

                }

                //bail so the plugin activity does not load...
                return;

        }else{
                if(dataURL.length() > 1){

                        //open default Android activity...
                        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(dataURL));
```

```
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
             startActivity(intent);

        }
    }
}

//END CHECK FOR PDF DOC
//////////////////////////////////////////////////////////////
```