Understanding App Refresh and Offline Apps

v1.0

By GoNorthWest

22 December 2011

One of the seriously cool features of an app built using Buzztouch is that you have the option of pushing out updates to the app **without** having to provide an updated app to iTunes or an Android market! What makes this possible is that all the information regarding the screen of an app – the type, name, data associated with the screen, etc – is contained in a file called **BT_config.txt**, which you can find in the BT_Config folder of your project. This file is contained in the final app package that goes to your users, and is referenced each time the app is opened, or brought back to the foreground after having been moved away due to another app opening.

The BT_config file contains a lot of information, but most important to our discussion is the field called **dataURL**, which can be found near the top of the file. It will look something like this:

```
"dataURL":http://www.buzztouch.com/appdata/v1.5.php?command=getAppData
&appGuid=[buzztouchAppId]&apiKey=[buzztouchAPIKey]
```

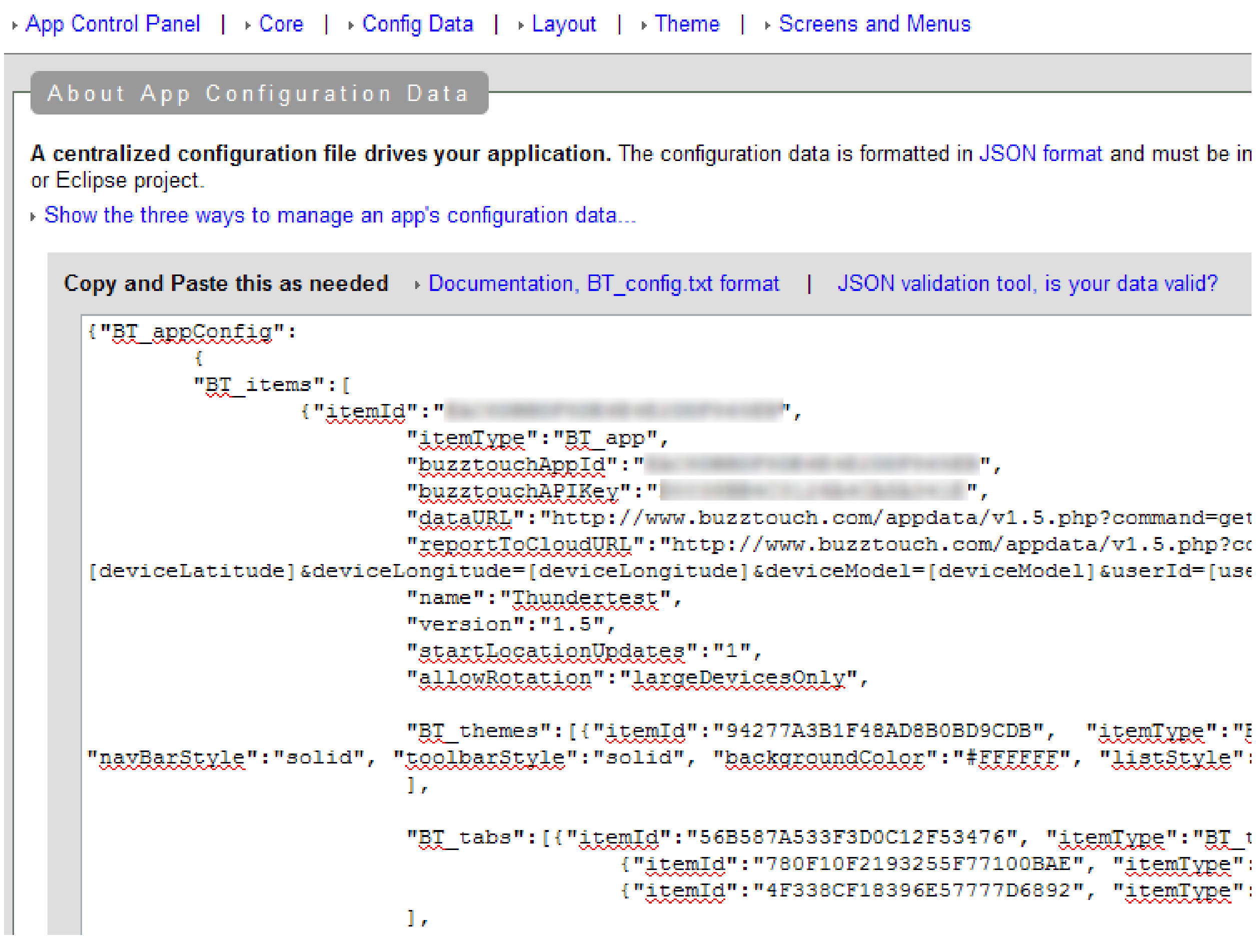
In a nutshell, if you have something in this field, then your app is considered "**online**," and it makes a call to that link each time the app is started or brought back to the forefront. If this field is blank, then your app is considered "**offline**," and it does not check the Internet for a current config file, but always uses the one that it ships with.

Does that make sense?

The dataURL can point to any place that has a BT_config.txt file in valid JSON format. In the default example above, it creates a URL based on your appGuid and apiKey (which are stored in the app) and retrieves the current configuration file from that location (it's actually generated on the fly). That URL could easily also point to a location like http://www.yourdomain.com/BT_Confix.txt, and as long as the file is valid JSON, all is good. This is handy if you want to store your config file off the Buzztouch servers, to reduce dependency on their service.

Let's take a moment and talk about the BT_config.txt file. As I said earlier, it contains all the information needed to build the screens of your app, and it written in JSON format. JSON stands for "JavaScript Object Notation," and is basically a way for exchanging data. There are rules on how the data must be formatted, and you can validate a JSON file by running it through JSONLint.com, which will check your format for you. For more information about JSON, check out http://en.wikipedia.org/wiki/JSON.

Each time you make a change to your application using the app control panel at Buzztouch.com, you make a change to your BT_config file. That information is stored on the Buzztouch servers, and is updated in realtime. If you ever want to see what your current configuration file looks like, just go to your apps control panel and select “Show App's Configuration Data” from the left side menu. You’ll then see a screen that looks something like this (hey, had to include at least one screenshot in this doc!):


App Control Panel | Core | Config Data | Layout | Theme | Screens and Menus
About App Configuration Data

A centralized configuration file drives your application. The configuration data is formatted in JSON format and must be in
or Eclipse project.

Show the three ways to manage an app's configuration data...

Copy and Paste this as needed Documentation, BT_config.txt format | JSON validation tool, is your data valid?

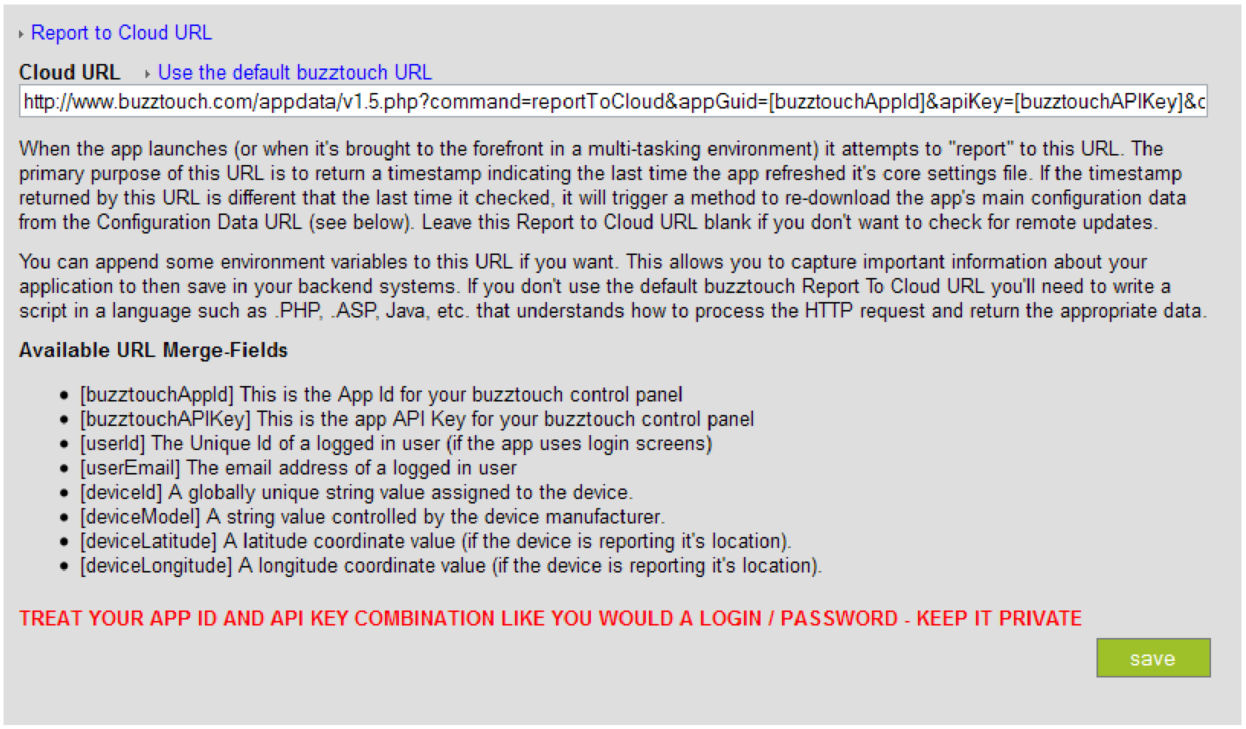
```
{"BT_appConfig":
        {
        "BT_items":[
                {"itemId":"[redacted]",
                        "itemType":"BT_app",
                        "buzztouchAppId":"[redacted]",
                        "buzztouchAPIKey":"[redacted]",
                        "dataURL":"http://www.buzztouch.com/appdata/v1.5.php?command=get
                        "reportToCloudURL":"http://www.buzztouch.com/appdata/v1.5.php?cc
[deviceLatitude]&deviceLongitude=[deviceLongitude]&deviceModel=[deviceModel]&userId=[use
                        "name":"Thundertest",
                        "version":"1.5",
                        "startLocationUpdates":"1",
                        "allowRotation":"largeDevicesOnly",

                        "BT_themes":[{"itemId":"94277A3B1F48AD8B0BD9CDB",  "itemType":"E
"navBarStyle":"solid", "toolbarStyle":"solid", "backgroundColor":"#FFFFFF", "listStyle":
                        ],

                        "BT_tabs":[{"itemId":"56B587A533F3D0C12F53476", "itemType":"BT_t
                                {"itemId":"780F10F2193255F77100BAE", "itemType":
                                {"itemId":"4F338CF18396E57777D6892", "itemType":
                        ],
```

When you download your app code package from Buzztouch, and look in BT_Docs\BT_config.txt, you will see the same information. Same file, different locations!

Now, there’s something to keep in mind here that seems obvious, but it may not be. If you make a change to your app using the control panel, it will update your online config file. So, at that moment, if your app is live in the wild, and it is configured with a dataURL that points to the Buzztouch servers, the next time somebody uses your app, they will be **prompted for a refresh**. That’s OK if you want to push out that new content, but perhaps not OK if you are testing new features. Now, you may be tempted to turn off the datURL, in essence removing the link, while you test new features, but that will take your app completely offline! If there is no link in the dataURL field, the next time somebody refreshes your app, it will take them completely offline, and only an update to the app itself (as in iTunes or marketplace) can put it back online. I’ll explain more about that shortly. The basic idea here is that if you are using the default dataURL for your app, and your app is live, then any configuration changes you make in the control panel will be pushed out to your users. Keep that in mind as you plan on future versions and features of your app.


Report to Cloud URL

Cloud URL Use the default buzztouch URL

http://www.buzztouch.com/appdata/v1.5.php?command=reportToCloud&appGuid=[buzztouchAppId]&apiKey=[buzztouchAPIKey]&c

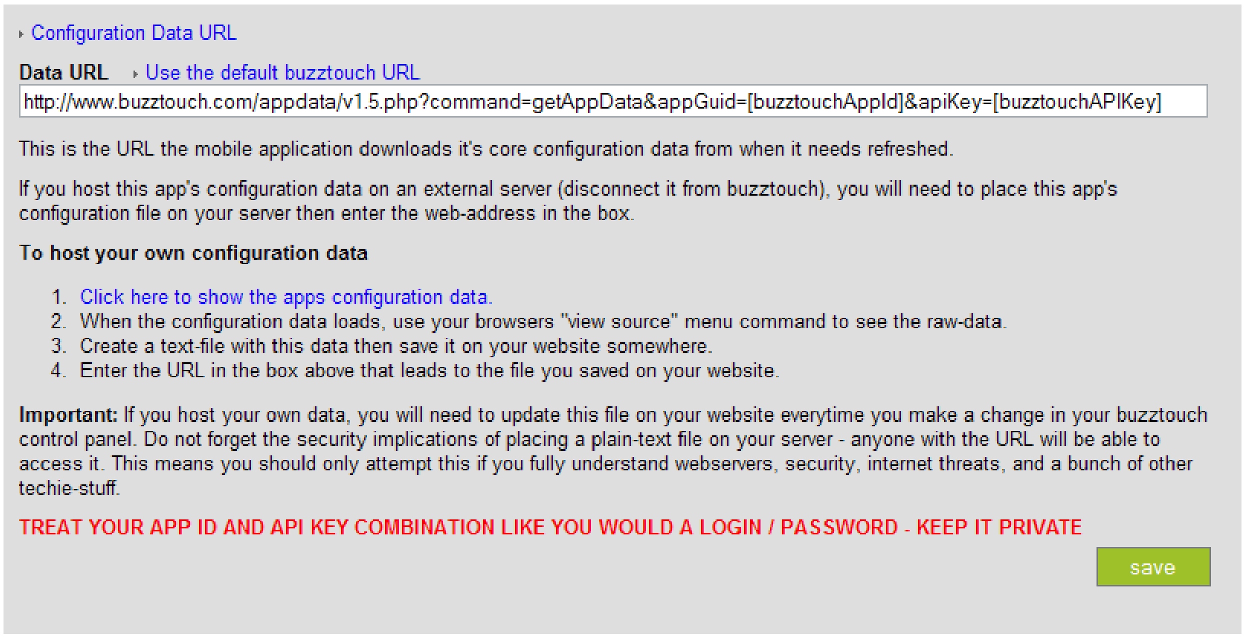
When the app launches (or when it's brought to the forefront in a multi-tasking environment) it attempts to "report" to this URL. The primary purpose of this URL is to return a timestamp indicating the last time the app refreshed it's core settings file. If the timestamp returned by this URL is different that the last time it checked, it will trigger a method to re-download the app's main configuration data from the Configuration Data URL (see below). Leave this Report to Cloud URL blank if you don't want to check for remote updates.

You can append some environment variables to this URL if you want. This allows you to capture important information about your application to then save in your backend systems. If you don't use the default buzztouch Report To Cloud URL you'll need to write a script in a language such as .PHP, .ASP, Java, etc. that understands how to process the HTTP request and return the appropriate data.

Available URL Merge-Fields

- [buzztouchAppId] This is the App Id for your buzztouch control panel
- [buzztouchAPIKey] This is the app API Key for your buzztouch control panel
- [userId] The Unique Id of a logged in user (if the app uses login screens)
- [userEmail] The email address of a logged in user
- [deviceId] A globally unique string value assigned to the device.
- [deviceModel] A string value controlled by the device manufacturer.
- [deviceLatitude] A latitude coordinate value (if the device is reporting it's location).
- [deviceLongitude] A longitude coordinate value (if the device is reporting it's location).

TREAT YOUR APP ID AND API KEY COMBINATION LIKE YOU WOULD A LOGIN / PASSWORD - KEEP IT PRIVATE


Configuration Data URL

Data URL Use the default buzztouch URL

http://www.buzztouch.com/appdata/v1.5.php?command=getAppData&appGuid=[buzztouchAppId]&apiKey=[buzztouchAPIKey]

This is the URL the mobile application downloads it's core configuration data from when it needs refreshed.

If you host this app's configuration data on an external server (disconnect it from buzztouch), you will need to place this app's configuration file on your server then enter the web-address in the box.

To host your own configuration data

1. Click here to show the apps configuration data.
2. When the configuration data loads, use your browsers "view source" menu command to see the raw-data.
3. Create a text-file with this data then save it on your website somewhere.
4. Enter the URL in the box above that leads to the file you saved on your website.

Important: If you host your own data, you will need to update this file on your website everytime you make a change in your buzztouch control panel. Do not forget the security implications of placing a plain-text file on your server - anyone with the URL will be able to access it. This means you should only attempt this if you fully understand webservers, security, internet threats, and a bunch of other techie-stuff.

TREAT YOUR APP ID AND API KEY COMBINATION LIKE YOU WOULD A LOGIN / PASSWORD - KEEP IT PRIVATE

Before we move on to different scenarios you may want to implement, there is one more Core Property, reportToCloudURL, that you should be aware of. Essentially, when an app is launched, or brought to the forefront in a multi-tasking environment, it attempts to "report" to URL specified in this field. The primary purpose of this URL is to trigger a refresh, and it does so by comparing timestamps of the current time with the last time this URL was reported to and a refresh indicated. If it is different than the last time it was checked, that will trigger a refresh, and the app will contact the URL specified in the dataURL field (or won't, if that field is blank). If the timestamps are the same, which means nothing has changed since the last refresh, then nothing happens. This URL can also be configured to report various items about your application, like what device is using it, when and where. Good stuff to know if you want to track that info. The takeaway item for this field is that if you leave it blank, the app won't attempt a refresh at all. If you leave it the default, it will attempt to check for a refresh, and will refresh using the dataURL link. If there is no dataURL link, then there will be no refresh. Got it? ;-)

OK, how about we take a look at some possible scenarios, and show you how to implement them? Cool? Cool!

Normal Refresh Operation (Online App). This mode doesn't touch any of the default settings. Both dataURL and reportToCloudURL are left untouched in the control panel.

- **How to do it:** Don't change anything in your app control panel. Leave all default values for Cloud URL and Data URL in the Core Properties section.
- **What to expect:** When the user opens the app, or brings the app to the forefront in a multi-tasking environment, the app will check for updates, and prompt the user to refresh if there are any.
- **Things to consider:** If you make any changes to the app in your control panel (like adding new screens, tweaking old ones, etc), then those will get pushed out to your users whether you want them to or not. So, testing new items is difficult, because people may get a refresh with something you are not ready to roll out. If you plan on continuing to develop your app with updates after you initially release it, consider a separate test app for new features that you can then port over to your current released app.

Suppress Refresh Operation (Offline App). This mode takes your app offline, meaning that users will not get the option to refresh their app based on changes you have made to it in the control panel.

- **How to do it:** There are actually two ways to accomplish this, both in the Core Properties section of your control panel.
    1. **Remove the Cloud URL link.** This will prevent the app from connecting to the Buzztouch servers to see when the last time the app was updated. It will also remove the Refresh button from the first screen of the app.
    2. **Remove the Data URL link.** This will prevent the app from downloading any new content, because there is no link it can follow to obtain a new BT_config file. If you make this blank, and leave the Cloud URL the default, it still puts the app offline, though it silently checks whether it's updated itself or not.
- **What to expect.** The user will not get any updates to the app in the standard way, and you will have to provide those updates via a full-fledged app update from either iTunes or an Android market. This cuts the app off from receiving updates via the Internet.

- **Things to consider. You cannot reverse this via the control panel, only by an update to the app in the store.** Once you have taken an app offline, the only way to get it back online is with a new version of the app that enables the feature. Don't make this mistake of turning this off while you test new features, as anybody who connects during that time will have their app taken offline!

Those are the two major ways that people want their apps to act, in an online or offline manner. Before we wrap this up, let's discuss a couple last points:

- Regardless of whether your app is online or offline, if you have screens that rely on a call to the Internet for content (like a Custom HTML screen, or a menu list populated by JSON data), that feature will still work. Taking an app offline does not disable the ability to make Internet calls for data from **within** the app.

- Neither the Data URL nor the Cloud URL have to point to the Buzztouch servers. You can host a BT_config file anywhere, and point the Data URL to it. In fact…that's a good way to test new features in your live app. Just put your current configuration file somewhere on the Internet (like the public folder in DropBox), change the Data URL to point to it, do your testing, and then point the Data URL back to the "new" configuration file! You can point those URLs to file or scripts or whatever, as long as they are in an acceptable JSON format.

- For your app to be completely, 100% offline, with no connection to the Internet required (meaning it would work fully in Airplane Mode), all your content needs to be local on the device. Nothing can point to a URL or anywhere off the device. Every screen or file will have to be packaged up with the app. Any updates will have to go through an update to the app.

I think that about sums it up! A very lengthy explanation of something that was kinda hard for me to wrap my head around, but made sense once I thought about it. A couple of people in the forum lost control of their app refresh (accidentally put them offline) because they didn't understand these concepts. Hopefully this will prevent that from happening to you, and help you configure your app in the way you really want it to work.

Good luck with your app! Happy coding!

Comments? Post them in the forum or email me at MrkFleming@gmail.com.

Revision Log			
v1.0	12/22/11	Initial release of document.	GoNorthWest