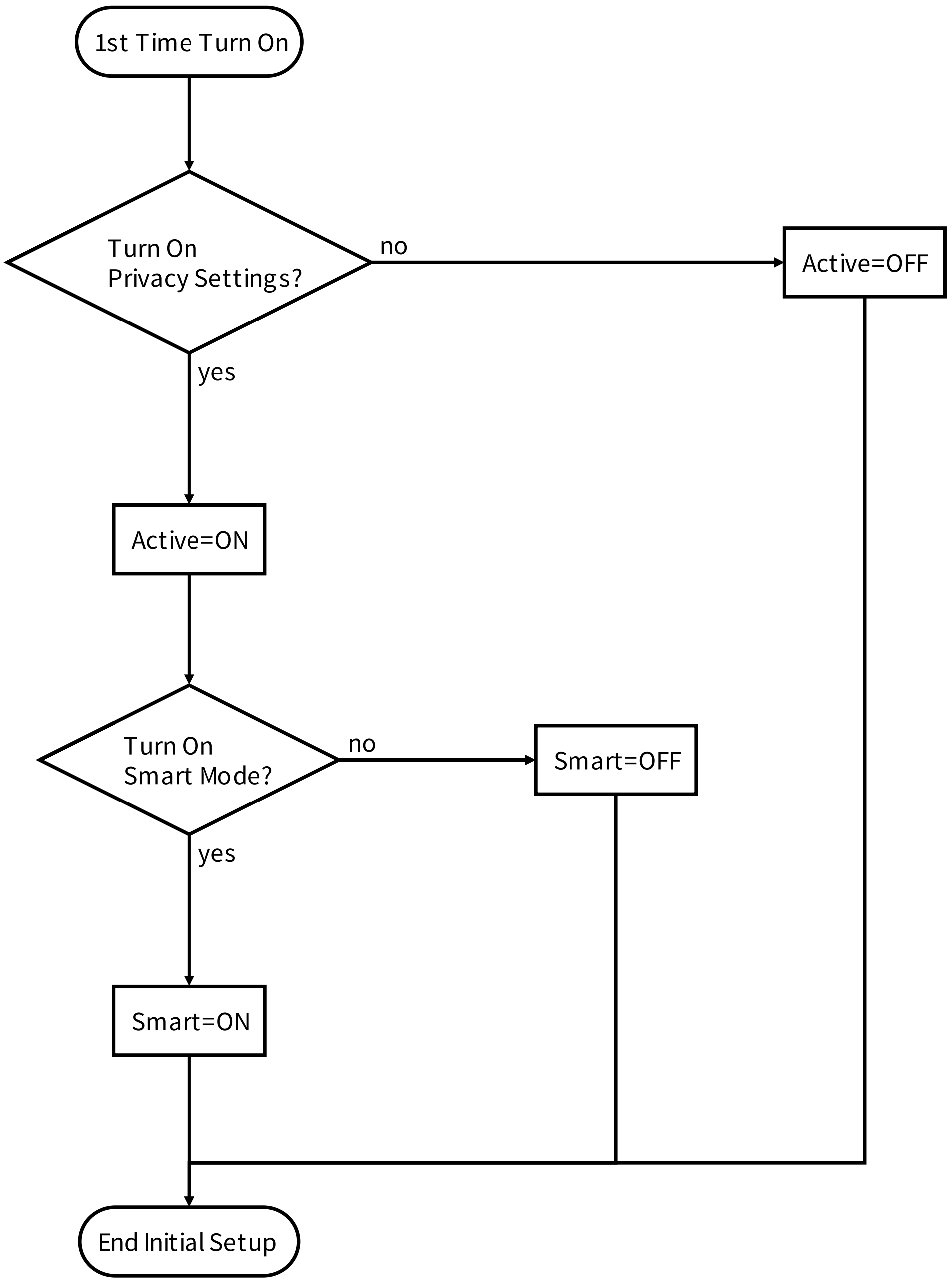
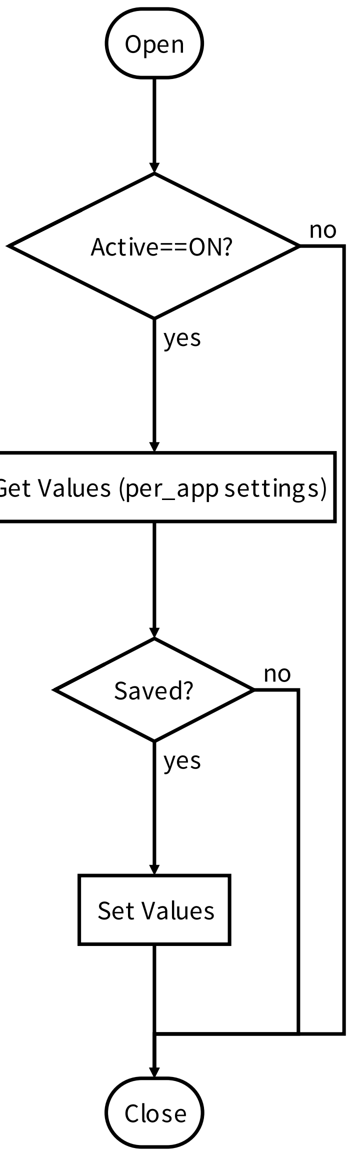
WEXposed Shared Preference Definitions

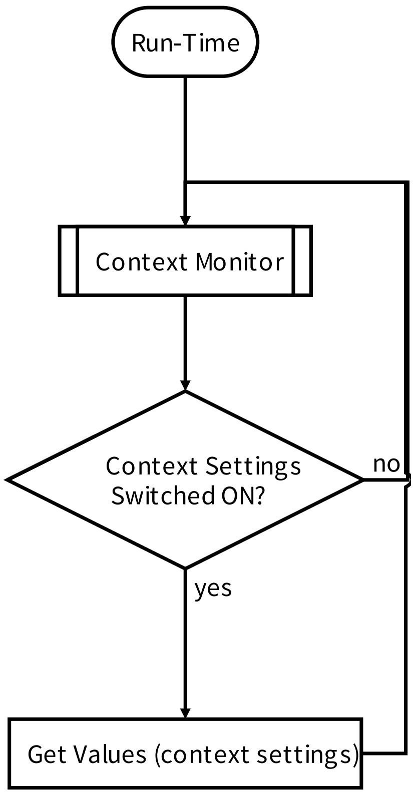
Initial Setup Process (Default Settings)



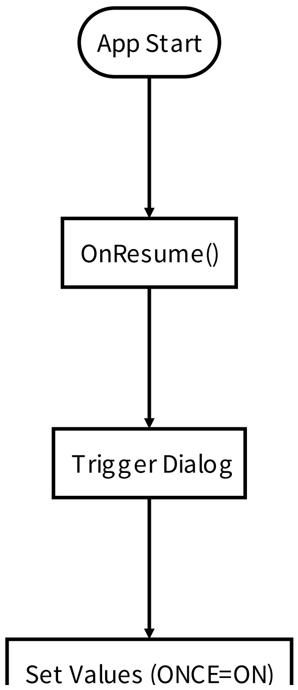
Per App Setup Process (User Defined Settings)

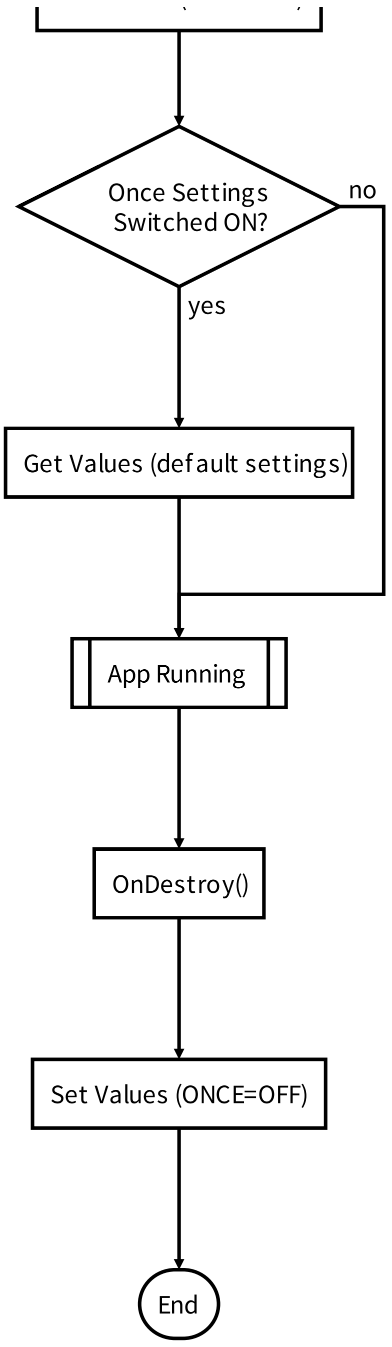


Context Settings



Once Settings





common.java

common.java is a public class containing static variables shared by both front-end and back-end. The variable names should be kept **EXACTLY AS FOLLOWS** while the values may be changed if necessary.

> **Note**: keep all the unused static variables for features to be added in the future.

```
package com.samsung.wexposed;

public class Common {

    public static final String TAG = "WEXposed";
    public static final String MY_PACKAGE_NAME =
Common.class.getPackage().getName();  //"com.samsung.wexposed"

    public static final String ACTION_PERMISSIONS = "update_permissions";

    public static final String PREFS = "PrivacySettings";  //setting file
name

    public static final String PREF_DEFAULT = "default";  //default settings
prefix
    public static final String PREF_WORK = "/work";  //work context prefix
    public static final String PREF_HOME = "/home";  //home context prefix
    public static final String PREF_GYM = "/gym";  //gym context prefix
    public static final String PREF_OUTDOOR = "/outdoor";  //outdoor
context prefix

    public static final String PREF_ACTIVE = "/active";  //active settings
per app
    public static final String PREF_CONTEXT = "/context";  //context
settings prefix per app
    public static final String PREF_CONTEXT = "/smart";  //smart mode prefix
per app
    public static final String PREF_ONCE = "/once";  //once settings prefix
per app
    public static final String PREF_APP = "/app";  //app settings prefix
    public static final String PREF_ADS = "/ads";  //ads settings prefix

    public static final String PREF_INTERNET = "/internet";  //internet
setting
    public static final String PREF_LOCATION = "/location";  //location
setting
    public static final String PREF_CONTACTS = "/contacts";  //contacts
setting
    public static final String PREF_SENSOR = "/sensor";  //sensor setting
}
```

The following function will be called when Zygote is initiated.

```
import de.robv.android.xposed.XSharedPreferences;

public static void loadPrefs() {
        prefs = new XSharedPreferences(Common.MY_PACKAGE_NAME,
Common.PREFS);
        prefs.makeWorldReadable();  //allow read from other apps.
}
```

PrivacySettings.xml

The shared preferences are stored in the following file.

```
/data/data/com.samsung.wexposed/shared_prefs/PrivacySettings.xml
```

A typical *PrivacySettings.xml* file looks like this.

```xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <boolean name="com.accuweather.android/active" value="true" />
    <boolean name="com.accuweather.android/once/active" value="false" />
    <boolean name="com.accuweather.android/smart/active" value="false" />
    <boolean name="com.accuweather.android/context/active" value="false" />

    <boolean name="com.accuweather.android/app/internet" value="true" />
    <boolean name="com.accuweather.android/ads/sensor" value="false" />
    <!-- Presets exist along with WEXposed -->
    <boolean name="com.accuweather.android/context/home/app/location"
value="false" />
    <boolean name="com.accuweather.android/context/home/app/sensor"
value="true" />
    <boolean name="com.accuweather.android/context/outdoor/app/location"
value="true" />
    <boolean name="com.accuweather.android/context/outdoor/app/sensor"
value="false" />
    <boolean name="default/ads/location" value="false" />
    <boolean name="default/app/internet" value="true" />
</map>
```

Note:
1. “com.accuweather.android/active” is the switch of *ALL* settings for *com.accuweather.android*. When switched on the per_app settings will be applied. If the per_app settings not defined, the default settings will be applied.
2. “com.accuweather.android/once/active” is the switch of *ONCE* settings for *com.accuweather.android*. When switched on the default settings will be applied.
3. “com.accuweather.android/context/active” is the switch of *CONTEXT* settings for *com.accuweather.android*. When switched on the presets associated with current context will be applied.

Get Values

Here follows an example of getting values from PrivacySetting.xml.

```
// Update Switch for Ads Location
((Switch)
findViewById(R.id.sw_ads_location)).setChecked(prefs.getBoolean(pkgName +
Common.PREF_ADS_LOCATION, false));
```

> **Convention:** use lowercase and “sw_”prefixes for Switch component ids on the UI.

Set Values

Here follows an example of setting values in PrivacySetting.xml.

```
...
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // if the SAVE button is clicked, update the values in shared preference
    if (item.getItemId() == R.id.menu_save) {
        Editor prefsEditor = prefs.edit();
        Map<String, Object> newSettings = getSettings();
        for (String key : settingKeys) {
            Object value = newSettings.get(key);
            if (value == null) {
                prefsEditor.remove(key);
            } else {
                if (value instanceof Boolean) {
                    prefsEditor.putBoolean(key, ((Boolean)
value).booleanValue());
                } else if (value instanceof Integer) {
                    prefsEditor.putInt(key, ((Integer) value).intValue());
                } else if (value instanceof String) {
                    prefsEditor.putString(key, (String) value);
                } else if (value instanceof Set) {
                    prefsEditor.remove(key);
                    // Commit and reopen the editor, as it seems to be
                    // bugged when updating a StringSet
                    prefsEditor.commit();
                    prefsEditor = prefs.edit();
                    prefsEditor.putStringSet(key, (Set<String>) value);
                } else {
                    // Should never happen
                    throw new IllegalStateException("Invalid setting type: "
+ key + "=" + value);
                }
            }
        }
        prefsEditor.commit();

        // Update saved settings to detect modifications later
        initialSettings = newSettings;
        ...
```

Hook Method

Here follows an example of hooking location function in WEXposed following the policy defined in the shared preference (i.e. PrivacySettings.xml).

```
loadPrefs();
// Hook to intercept ads location
findAndHookMethod("android.location.LocationManager", lpparam.classLoader,
"getLastKnownLocation", String.class, new XC_MethodHook() {
    @Override
    protected void afterHookedMethod(MethodHookParam param) throws Throwable
{
        String packageName = AndroidAppHelper.currentPackageName();

        if (prefs.getBoolean(packageName + Common.PREF_ACTIVE, false) {
            // No overrides for this package
            return;
        }

        boolean appLocation = true;  //allow location by default
        // if Once Settings are switched ON, use default settings
        if(prefs.getBoolean(Common.PREF_ONCE + Common.PREF_ACTIVE, false){
            appLocation = prefs.getBoolean(Common.PREF_DEFAULT +
Common.PREF_ADS + Common.PREF_LOCATION, false);
        }

        // if context Settings are switched ON, use default settings
        else if(prefs.getBoolean(packageName + Common.PREF_CONTEXT +
Common.PREF_ACTIVE, false){
            appLocation = prefs.getBoolean(packageName + Common.PREF_CONTEXT
+ Common.PREF_ADS + Common.PREF_LOCATION,
prefs.getBoolean(Common.PREF_DEFAULT + Common.PREF_ADS +
Common.PREF_LOCATION, false));
```

```
        }

        // use per_app settings
        else{
            appLocation = prefs.getBoolean(packageName + Common.PREF_ADS +
Common.PREF_LOCATION, prefs.getBoolean(Common.PREF_DEFAULT +
Common.PREF_ADS + Common.PREF_LOCATION, false));
        }

        if (!appLocation) {
            ...
        }
    };
});
```

> **Note:** The priority sequence of the settings is: once > context > per_app > default. (i.e.: “>” denotes override).