Using the Backup API

When a user purchases a new device or resets their existing one, they might expect that when Google Play restores your app back to their device during the initial setup, the previous data associated with the app restores as well. By default, that doesn't happen and all the user's accomplishments or settings in your app are lost.

For situations where the volume of data is relatively light (less than a megabyte), like the user's preferences, notes, game high scores or other stats, the Backup API provides a lightweight solution. This lesson walks you through integrating the Backup API into your application, and restoring data to new devices using the Backup API.

THIS LESSON TEACHES YOU TO

1. Register for the Android Backup Service
2. Configure Your Manifest
3. Write Your Backup Agent
4. Request a Backup
5. Restore from a Backup

YOU SHOULD ALSO READ

- Data Backup

Register for the Android Backup Service

This lesson requires the use of the Android Backup Service (http://code.google.com/android/backup/index.html), which requires registration. Go ahead and register here (http://code.google.com/android/backup/signup.html). Once that's done, the service pre-populates an XML tag for insertion in your Android Manifest, which looks like this:

```
<meta-data android:name="com.google.android.backup.api_key"
android:value="ABcDe1FGHij2KlmN3oPQRs4TUvW5xYZ" />
```

Note that each backup key works with a specific package name. If you have different applications, register separate keys for each one.

Configure Your Manifest

Use of the Android Backup Service requires two additions to your application manifest. First, declare the name of the class that acts as your backup agent, then add the snippet above as a child element of the Application tag. Assuming your backup agent is going to be called `TheBackupAgent`, here's an example of what the manifest looks like with this tag included:

```
<application android:label="MyApp"
             android:backupAgent="TheBackupAgent">
    ...
    <meta-data android:name="com.google.android.backup.api_key"
    android:value="ABcDe1FGHij2KlmN3oPQRs4TUvW5xYZ" />
    ...
</application>
```

Write Your Backup Agent

The easiest way to create your backup agent is by extending the wrapper class `BackupAgentHelper` (../../reference/android/app/backup/BackupAgentHelper.html). Creating this helper class is actually a very simple process. Just create a class with the same name as you used in the manifest in the previous step (in this example, `TheBackupAgent`), and extend `BackupAgentHelper`. Then override the `onCreate()` (../../reference/android/app/backup/BackupAgent.html#onCreate()).

Inside the `onCreate()` (../../reference/android/app/backup/BackupAgent.html#onCreate()) method, create a `BackupHelper` (../../reference/android/app/backup/BackupHelper.html). These helpers are specialized classes for backing up certain kinds of data. The Android framework currently includes two such helpers: `FileBackupHelper` (../../reference/android/app/backup/FileBackupHelper.html) and `SharedPreferencesBackupHelper` (../../reference/android/app/backup/SharedPreferencesBackupHelper.html). After you create the helper and point it at the data you want to back up, just add it to the BackupAgentHelper using the `addHelper()` (../../reference/android/app/backup/BackupAgentHelper.html#addHelper(java.lang.String, android.app.backup.BackupHelper)) method, adding a key which is used to retrieve the data later. In most cases the entire implementation is perhaps 10 lines of code.

Here's an example that backs up a high scores file.

```
import android.app.backup.BackupAgentHelper;
import android.app.backup.FileBackupHelper;


public class TheBackupAgent extends BackupAgentHelper {
    // The name of the SharedPreferences file
    static final String HIGH_SCORES_FILENAME = "scores";

    // A key to uniquely identify the set of backup data
    static final String FILES_BACKUP_KEY = "myfiles";

    // Allocate a helper and add it to the backup agent
    @Override
    void onCreate() {
        FileBackupHelper helper = new FileBackupHelper(this, HIGH_SCORES_FILENAME);
        addHelper(FILES_BACKUP_KEY, helper);
    }
}
```

For added flexibility, `FileBackupHelper` (../../reference/android/app/backup/FileBackupHelper.html)'s constructor can take a variable number of filenames. You could just as easily have backed up both a high scores file and a game progress file just by adding an extra parameter, like this:

```
    @Override
    void onCreate() {
        FileBackupHelper helper = new FileBackupHelper(this, HIGH_SCORES_FILENAME,
        addHelper(FILES_BACKUP_KEY, helper);
    }
```

Backing up preferences is similarly easy. Create a `SharedPreferencesBackupHelper` (../../reference/android/app/backup/SharedPreferencesBackupHelper.html) the same way you did a

FileBackupHelper (../../reference/android/app/backup/FileBackupHelper.html). In this case, instead of adding filenames to the constructor, add the names of the shared preference groups being used by your application. Here's an example of how your backup agent helper might look if high scores are implemented as preferences instead of a flat file:

```
import android.app.backup.BackupAgentHelper;
import android.app.backup.SharedPreferencesBackupHelper;

public class TheBackupAgent extends BackupAgentHelper {
    // The names of the SharedPreferences groups that the application maintains.
    // are the same strings that are passed to getSharedPreferences(String, int).
    static final String PREFS_DISPLAY = "displayprefs";
    static final String PREFS_SCORES = "highscores";

    // An arbitrary string used within the BackupAgentHelper implementation to
    // identify the SharedPreferencesBackupHelper's data.
    static final String MY_PREFS_BACKUP_KEY = "myprefs";

    // Simply allocate a helper and install it
    void onCreate() {
        SharedPreferencesBackupHelper helper =
                new SharedPreferencesBackupHelper(this, PREFS_DISPLAY, PREFS_SCOR
        addHelper(MY_PREFS_BACKUP_KEY, helper);
    }
}
```

You can add as many backup helper instances to your backup agent helper as you like, but remember that you only need one of each type. One FileBackupHelper (../../reference/android/app/backup/FileBackupHelper.html) handles all the files that you need to back up, and one SharedPreferencesBackupHelper (../../reference/android/app/backup/SharedPreferencesBackupHelper.html) handles all the shared preferencegroups you need backed up.

Request a Backup

In order to request a backup, just create an instance of the BackupManager (../../reference/android/app/backup/BackupManager.html), and call it's dataChanged() (../../reference/android/app/backup/BackupManager.html#dataChanged()) method.

```
import android.app.backup.BackupManager;
...

public void requestBackup() {
  BackupManager bm = new BackupManager(this);
  bm.dataChanged();
}
```

This call notifies the backup manager that there is data ready to be backed up to the cloud. At some point in the future, the backup manager then calls your backup agent's onBackup()

(../../reference/android/app/backup/BackupAgent.html#onBackup(android.os.ParcelFileDescriptor, android.app.backup.BackupDataOutput, android.os.ParcelFileDescriptor)) method. You can make the call whenever your data has changed, without having to worry about causing excessive network activity. If you request a backup twice before a backup occurs, the backup only occurs once.

Restore from a Backup

Typically you shouldn't ever have to manually request a restore, as it happens automatically when your application is installed on a device. However, if it *is* necessary to trigger a manual restore, just call the `requestRestore()` (../../reference/android/app/backup/BackupManager.html#requestRestore(android.app.backup.RestoreObserver)) method.