BatteryStasService的主要功能是收集系统中各模块和应用进程的用电情况。因此，我们可以认为BatteryStatsService是Android中的“电表”。

只不过这个电表比较智能，不是单纯地统计整体的耗电，而是分门别类的统计每个部分的耗电情况。接下来我们就分析一下BatteryStatsService的主要流程。为了方便叙述，后文中我们将BatteryStatsService简称为BSS。我们这次的参考代码为Android P

~/frameworks/base/services/core/java/com/android/server/am/BatteryStatsService.java

### 构造函数

与一般的系统服务不太一样，BSS的创建和发布是在ActivityManagerService中进行的，相关代码如下：

~/frameworks/base/services/core/java/com/android/server/am/ActivityManagerService.java

public ActivityManagerService(Context systemContext) {

...

File dataDir = Environment.getDataDirectory();

File systemDir = new File(dataDir, "system");

systemDir.mkdirs();

…

//创建BSS对象，传入/data/system目录，同时传入ActivityManagerService的handler

mBatteryStatsService = new BatteryStatsService(systemContext, systemDir, mHandler);

//调用BSS中BatteryStatsImpl对象的readLocked方法

mBatteryStatsService.getActiveStatistics().readLocked();

//将初始化得到的信息写入disk

mBatteryStatsService.scheduleWriteToDisk();

...

}

接下来我们先看看BatteryStatsService的构造函数：

BatteryStatsService(Context context, File systemDir, Handler handler) {//传递的参数：文件路径/data/system，AMS MainHandler

//BatteryStatsImpl expects the ActivityManagerService handler, so pass that one through.

mContext = context;

mUserManagerUserInfoProvider = new BatteryStatsImpl.UserInfoProvider() {

private UserManagerInternal umi;

@Override

public int[] getUserIds() {

if (umi == null) {

umi = LocalServices.getService(UserManagerInternal.class);

}

return (umi != null) ? umi.getUserIds() : null;

}

};

mStats = new BatteryStatsImpl(systemDir, handler, this, mUserManagerUserInfoProvider);

mWorker = new BatteryExternalStatsWorker(context, mStats);

mStats.setExternalStatsSyncLocked(mWorker);

mStats.setRadioScanningTimeoutLocked(mContext.getResources().getInteger(com.android.internal.R.integer.config\_radioScanningTimeout) \* 1000L);

mStats.setPowerProfileLocked(new PowerProfile(context));

}