Monitoring Data Connectivity and Activity

As well as voice and service details, you can monitor changes in mobile data connectivity and mobile data transfer by implementing a PhoneStateListener.

The Phone State Listener includes two event handlers for monitoring the device data connection. Override onDataActivity to track data transfer activity, and onDataConnectionStateChanged to request notifi cations for data connection state changes.

The following skeleton code shows both handlers overridden, with switch statements demonstrating each of the possible values for the state and direction parameters passed in to each event:

```
PhoneStateListener dataStateListener = new PhoneStateListener() {
public void onDataActivity(int direction) {
switch (direction) {
case TelephonyManager.DATA_ACTIVITY_IN : break;
case TelephonyManager.DATA_ACTIVITY_OUT : break;
case TelephonyManager.DATA_ACTIVITY_INOUT : break;
case TelephonyManager.DATA_ACTIVITY_NONE : break;
}
}
public void onDataConnectionStateChanged(int state) {

switch (state) {
case TelephonyManager.DATA_CONNECTED : break;
case TelephonyManager.DATA_CONNECTING : break;
case TelephonyManager.DATA_DISCONNECTED : break;
case TelephonyManager.DATA_SUSPENDED : break;
}
}
};
telephonyManager.listen(dataStateListener,
PhoneStateListener.LISTEN_DATA_ACTIVITY |
PhoneStateListener.LISTEN_DATA_CONNECTION_STATE);
```

Accessing Phone Properties and Status

The Telephony Manager also provides access to several static phone properties. You can obtain the current value of any of the phone state details described previously. The following code snippet shows how to extract the current incoming call number if the phone is ringing:

```
String incomingCall = null;
if (telephonyManager.getCallState() == TelephonyManager.CALL_STATE_RINGING)
incomingCall = telephonyManager.getLine1Number();
```

You can also access SIM and network operator details, network information, and voice-mail details. The following code snippet shows the framework used to access the current network details:

```
String srvcName = Context.TELEPHONY_SERVICE;
TelephonyManager telephonyManager = (TelephonyManager)getSystemService(srvcName);
String networkCountry = telephonyManager.getNetworkCountryIso();
String networkOperatorId = telephonyManager.getNetworkOperator();
String networkName = telephonyManager.getNetworkOperatorName();
int networkType = telephonyManager.getNetworkType();
```

Controlling the Phone

There are times when you need access to the underlying phone hardware to effect changes rather than simply monitoring them.

Access to the underlying Phone hardware was removed shortly before the release of Android SDK version 1. It is expected that basic phone interaction including answering and hanging up the phone will be available in a subsequent API release.

The following section is based on an earlier API release that included phone hardware interaction support. It has been included to serve as a guide for likely future implementations.

The Phone class in Android provides this interface, letting you control hardware settings, handle incoming calls, initiate new outgoing calls, hang up calls in progress, handle conference calls, and a variety of other Phone functionalities.

Replacing the basic phone functionality is a complex process. Rather than delve into this in detail, this section focuses on some of the more useful functions that can be particularly powerful when used within applications that augment rather than replace the existing functionality.

To access the phone hardware, use the Phone class, available through the Telephony Manager using the getPhone method as shown in the following code snippet:

```
Phone phone = telephonyManager.getPhone();
```

Once you have a reference to the Phone object, you can initiate calls using the call or dial method or end them by calling endCall.

Answering, Dismissing, and Ending Calls

The following code snippet shows how to use the Phone State Listener to listen for incoming calls and reject them if they're from a particular place:

```
final String badPrefix = "+234";
PhoneStateListener callBlockListener = new PhoneStateListener() {
public void onCallStateChanged(int state, String incomingNumber) {
if (state == TelephonyManager.CALL_STATE_RINGING) {
Phone phone = telephonyManager.getPhone();
if (incomingNumber.startsWith(badPrefix)) {
phone.endCall();
}
}
}
};
telephonyManager.listen(callBlockListener,
PhoneStateListener.LISTEN_CALL_STATE);
```