Appendix - Complete Examples

Library Module for iOS

```
library module com.livecode.platform.sensors.battery.ios

  use com.livecode.wrapped_apis_v1.livecode.objc
  use com.livecode.wrapped_apis_v1.apple.uikit.uidevice

  /* State querying functions */

  public handler isBatteryMonitoringEnabled() returns Boolean
    return UIDeviceGetBatteryMonitoringEnabled(UIDeviceGetCurrentDevice())
  end handler

  public handler isBatteryMonitoringDisabled() returns Boolean
    return not isBatteryMonitoringEnabled()
  end handler

  public handler batteryState() returns String
    variable tState as Integer
    put UIDeviceGetBatteryState(UIDeviceGetCurrentDevice()) into tState
    if tState is UIDeviceBatteryStateUnplugged then
      return "unplugged"
    else if tState is UIDeviceBatteryStateCharging then
      return "charging"
    else if tState is UIDeviceBatteryStateCharged then
      return "charged"
    end if
    return ""
  end handler

  public handler batteryLevel() returns optional Number
    variable tLevel as Number
    put UIDeviceGetBatteryLevel(UIDeviceGetCurrentDevice()) into tLevel
    if tLevel is -1.0 then
      return nothing
    end if
    return tLevel
  end handler
```

```
/* Service start and stop commmands */

variable mBatteryStateCookie as optional LiveCodeObjCNotificationCookie
variable mBatteryLevelCookie as optional LiveCodeObjCNotificationCookie

public handler startMonitoringBattery()
  if mBatteryStateCookie is not nothing then
    return
  end if
  LiveCodeObjCRegisterNotification(UIDeviceBatteryStateChanged, \
                handleBatteryStateChanged, \
                mBatteryStateCookie)
  LiveCodeObjCRegisterNotification(UIDeviceBatteryLevelChanged, \
                handleBatteryLevelChanged, \
                mBatteryLevelCookie)
  UIDeviceSetBatteryMonitoringEnabled(UIDeviceGetCurrentDevice(), true)
end handler

public handler stopMonitoringBattery()
  if mBatteryLevelCookie is nothing then
    return
  end if
  UIDeviceSetBatteryMonitoringEnabled(UIDeviceGetCurrentDevice(), false)
  LoveCodeCObjCDeregisterNotification(mBatteryStateCookie)
  put nothing into mBatteryStateCookie
  LiveCodeObjCDeregisterNotification(mBatteryLevelCookie)
  put nothing into mBatteryLevelCookie
end handler

private handler handleBatteryStateChanged(in pInfo as Pointer)
  variable tTargetObject as ScriptObject
  resolve script object "this card" into tTargetObject
  post "batteryStateChanged" to tTargetObject
end handler

private handler handleBatteryLevelChanged(in pInfo as Pointer)
  variable tTargetObject as ScriptObject
  resolve script object "this card" into tTargetObject
  post "batterySChanged" to tTargetObject
end handler
```

```
end module /* com.livecode.platform.sensors.battery.ios */
```

Foreign Module for iOS

```
foreign module com.livecode.wrapped_apis_v1.apple.uikit.uidevice

  use com.livecode.wrapped_apis_v1.apple.foundation.nsobject
  use com.livecode.wrapped_apis_v1.apple.foundation.nsuuid

  pointer type UIDevice is an NSObject

  /* Getting the Shared Device Instance */

  handler UIDeviceGetCurrentDevice() \
      returns UIDevice
    wrap objc UIDevice +(UIDevice *)currentDevice
    return result
  end handler

  /* Determining the Available Features */

  handler UIDeviceGetMultitaskingSupported(in pDevice as UIDevice) \
      returns Boolean
    wrap objc UIDevice -(BOOL)multitaskingSupported
    return result
  end handler

  /* Identifying the Device and Operating System */

  handler UIDeviceGetName(in pDevice as UIDevice) \
      returns String
    wrap objc UIDevice -(NSString *)name
    return result
  end handler

  handler UIDeviceGetSystemName(in pDevice as UIDevice) \
      returns String
    wrap objc UIDevice -(NSString *)systemName
    return result
  end handler

  handler UIDeviceGetSystemName(in pDevice as UIDevice) \
```

```
    returns String
  wrap objc UIDevice -(NSString *)systemName
  return result
end handler

handler UIDeviceGetSystemVersion(in pDevice as UIDevice) \
    returns String
  wrap objc UIDevice -(NSString *)systemVersion
  return result
end handler

handler UIDeviceGetModel(in pDevice as UIDevice) \
    returns String
  wrap objc UIDevice -(NSString *)model
  return result
end handler

handler UIDeviceGetLocalizedModel(in pDevice as UIDevice) \
    returns String
  wrap objc UIDevice -(NSString *)localizedModel
  return result
end handler

enum UIUserInterfaceIdiom
  case Unspecified is -1
  case Phone
  case Pad
  case TV
  case CarPlay
end enum

handler UIDeviceGetUserInterfaceIdiom(in pDevice as UIDevice) \
    returns UIUserInterfaceIdiom
  wrap objc UIDevice -(UIUserInterfaceIdiom)userInterfaceIdiom
  return result
end handler

handler UIDeviceGetIdentifierForVendor(in pDevice as UIDevice) \
    returns NSUUID
  wrap objc UIDevice -(NSUUID *)identifierForVendor
  return result
```

```
  end handler

  /* Getting the Device Orientation */

  enum UIDeviceOrientation
    case Unknown
    case Portrait
    case PortraitUpsideDown
    case LandscapeLeft
    case LandscapeRight
    case FaceUp
    case FaceDown
  end enum

  handler UIDeviceGetOrientation(in pDevice as UIDevice) \
      returns UIDeviceOrientation
    wrap objc UIDevice -(UIDeviceOrientation)orientation
    return result
  end handler

  handler UIDeviceGeneratesOrientationNotifications(in pDevice as UIDevice) \
      returns Boolean
    wrap objc UIDevice -(BOOL)generatedDeviceOrientationNotifications
    return result
  end handler

  handler UIDeviceBeginGeneratingDeviceOrientationNotifications(in pDevice as
UIDevice) \
      returns nothing
    wrap objc UIDevice -(void)beginGeneratingDeviceOrientationNotifications
    return nothing
  end handler

  handler UIDeviceEndGeneratingDeviceOrientationNotifications(in pDevice as
UIDevice) \
      returns nothing
    wrap objc UIDevice -(void)endGeneratingDeviceOrientationNotifications
    return nothing
  end handler

  notification UIDeviceOrientationDidChangeNotification
```

```
/* Getting the Device Battery State */

handler UIDeviceGetBatteryLevel(in pDevice as UIDevice) \
    returns Number
  wrap objc UIDevice -(float)batteryLevel
  return result
end handler

handler UIDeviceGetBatteryMonitoringEnabled(in pDevice as UIDevice) \
    returns Boolean
  wrap objc UIDevice -(BOOL)batteryMonitoringEnabled
  return result
end handler

handler UIDeviceSetBatteryMonitoringEnabled(in pDevice as UIDevice, \
                                    in pEnabled as Boolean) \
    returns nothing
  wrap objc UIDevice -(void)setBatteryMonitoringEnabled: (BOOL)enabled
  in pEnabled into enabled
  return nothing
end handler

enum UIDeviceBatteryState
  case Unknown
  case Unplugged
  case Charging
  case Full
end enum

handler UIDeviceGetBatteryState(in pDevice as UIDevice) \
    returns UIDeviceBatteryState
  wrap objc UIDevice -(UIDeviceBatteryState)batteryState
  return result
end handler

notification UIDeviceBatteryLevelDidChangeNotification
notification UIDeviceBatteryStateDidChangeNotification

/* Using the Proximity Sensor */
```

```
  handler UIDeviceGetProximityMonitoringEnabled(in pDevice as UIDevice) \
      returns Boolean
    wrap objc UIDevice -(BOOL)proximityMonitoringEnabled
    return result
  end handler

  handler UIDeviceSetProximityMonitoryEnabled(in pDevice as UIDevice, \
                                    in pEnabled as Boolean) \
      returns nothing
    wrap objc UIDevice -(void)setProximityMonitoringEnabled:(BOOL)enabled
    in pEnabled into enabled
    return nothing
  end handler

  handler UIDeviceGetProximityState(in pDevice as UIDevice) \
      returns Boolean
    wrap objc UIDevice -(BOOL)proximityState
    return result
  end handler

  notification UIDeviceProximityStateDidChangeNotification

end module /* com.livecode.wrapped_apis_v1.apple.uikit.uidevice */
```

Library Module for Android

```
library module com.livecode.platform.sensors.battery.android

  use com.livecode.wrapped_apis_v1.livecode.android
  use com.livecode.wrapped_apis_v1.android.content.intent
  use com.livecode.wrapped_apis_v1.android.os.batterymanager

  /* State querying functions */

  variable mBatteryChangedIntent as optional AndroidContentIntent
  variable mBatteryChangedCookie as optional LiveCodeAndroidReceiverCookie

  public handler isBatteryMonitoringEnabled() returns Boolean
    return mBatteryChangedCookie is not nothing
  end handler

  public handler isBatteryMonitoringDisabled() returns Boolean
```

```
    return not isBatteryMonitoringEnabled()
  end handler

  public handler batteryState() returns String
    if mBatteryChangedCookie is nothing then
      return ""
    end if

    variable tPresent as Boolean
    put AndroidContentIntentGetBooleanExtra(mBatteryChangedIntent, \

                                ANDROID_OS_BATTERY_MANAGER_EXTRA_PRE
SENT, \
                                true) \
                                into tPresent

    variable tPlugged as Integer
    put AndroidContentIntentGetIntExtra(mBatteryChangedIntent, \

                               ANDROID_OS_BATTERY_MANAGER_EXTRA_PLUG
GED,
                               -1) \
                               into tPlugged

    if not tPresent or tPlugged is -1 then
      return ""
    end if

    variable tState as Integer
    put AndroidContentIntentGetIntExtra(mBatteryChangedIntent, \

                               ANDROID_OS_BATTERY_MANAGER_EXTRA_STAT
US, \
                               -1) \
                               into tState

    if not tPresent or tPlugged is -1 then
      return ""
    end if

    if tState is
```

```
ANDROID_OS_BATTERY_MANAGER_BATTERY_STATUS_FULL then
      return "charged"
    end if

    if tPlugged is not -1 then
      return "charging"
    end if

    return "unplugged"
  end handler

  public handler batteryLevel() returns optional Number
    if mBatteryChangedCookie is nothing then
      return ""
    end if

    variable tCurrent as Integer
    put AndroidContentIntentGetIntExtra(mBatteryChangedIntent, \

                                ANDROID_OS_BATTERY_MANAGER_EXTRA_LEVE
L, \
                                -1) \
                                into tCurrent

    variable tMax as Integer
    put AndroidContentIntentGetIntExtra(mBatteryChangedIntent, \

                                ANDROID_OS_BATTERY_MANAGER_EXTRA_LEVE
L, \
                                -1) \
                                into tMax

    variable tLevel as Number
    put tCurrent / tMax into tLevel
    if tLevel < 0.0 or tLevel > 1.0 then
      put 1.0 into tLevel
    end if

    return tLevel
  end handler
```

```
  /* Service start and stop commmands */

  public handler startMonitoringBattery()
    if mBatteryChangedCookie is not nothing then
      return
    end if


    LiveCodeAndroidRegisterReceiver(AndroidContentIntent.ACTION_BATTERY_
CHANGED, \
                                handleBatteryChanged,
                                mBatteryChangedIntent, \
                                mBatteryChangedCookie)
  end handler

  public handler stopMonitoringBattery()
    if mBatteryChangedCookie is nothing then
      return
    end if

    LiveCodeAndroidDeregisterReceiver(mBatteryChangedIntent, \
                                  mBatteryChangedCookie)
    put nothing into mBatteryChangedIntent
    put nothing into mBatteryChangedCookie
  end handler

  private handler handleBatteryChanged(in pIntent as AndroidContentIntent)
    variable tTargetObject as ScriptObject
    resolve script object "this card" into tTargetObject
    post “batteryStateChanged” to tTargetObject

    variable tTargetObject as ScriptObject
    resolve script object "this card" into tTargetObject
    post “batteryLevelChanged” to tTargetObject
  end handler

end module /* com.livecode.platform.sensors.battery.android */
```

Foreign Module for Android

```
foreign module com.livecode.wrapped_apis_v1.android.os.batterymanager
```

use com.livecode.wrapped_apis_v1.java.lang.object

pointer type AndroidOsBatteryManager is a JavaLangObject

constant ANDROID_OS_BATTERY_MANAGER_ACTION_CHARGING is "android.os.action.CHARGING"
constant ANDROID_OS_BATTERY_MANAGER_ACTION_DISCHARGING is "android.os.action.DISCHARGING"

constant ANDROID_OS_BATTERY_MANAGER_BATTERY_HEALTH_COLD is 7
constant ANDROID_OS_BATTERY_MANAGER_BATTERY_HEALTH_DEAD is 4
constant ANDROID_OS_BATTERY_MANAGER_BATTERY_HEALTH_GOOD is 2
constant ANDROID_OS_BATTERY_MANAGER_BATTERY_HEALTH_OVERHEAT is 3
constant ANDROID_OS_BATTERY_MANAGER_BATTERY_HEALTH_OVER_VOLTAGE is 5
constant ANDROID_OS_BATTERY_MANAGER_BATTERY_HEALTH_UNKNOWN is 1
constant ANDROID_OS_BATTERY_MANAGER_BATTERY_HEALTH_UNSPECIFIED_FAILURE is 6

constant ANDROID_OS_BATTERY_MANAGER_BATTERY_PLUGGED_AC is 1
constant ANDROID_OS_BATTERY_MANAGER_BATTERY_PLUGGED_USB is 1
constant ANDROID_OS_BATTERY_MANAGER_BATTERY_PLUGGED_WIRELESS is 1

constant ANDROID_OS_BATTERY_MANAGER_BATTERY_PROPERTY_CAPACITY is 4
constant ANDROID_OS_BATTERY_MANAGER_BATTERY_PROPERTY_CHARGE_COUNTER is 1
constant

ANDROID_OS_BATTERY_MANAGER_BATTERY_PROPERTY_CURRENT_AVERAGE is 3
  constant ANDROID_OS_BATTERY_MANAGER_BATTERY_PROPERTY_CURRENT_NOW is 2
  constant ANDROID_OS_BATTERY_MANAGER_BATTERY_PROPERTY_ENERGY_COUNTER is 5

  constant ANDROID_OS_BATTERY_MANAGER_BATTERY_STATUS_CHARGING is 2
  constant ANDROID_OS_BATTERY_MANAGER_BATTERY_STATUS_DISCHARGING is 3
  constant ANDROID_OS_BATTERY_MANAGER_BATTERY_STATUS_FULL is 5
  constant ANDROID_OS_BATTERY_MANAGER_BATTERY_STATUS_NOT_CHARGING is 4
  constant ANDROID_OS_BATTERY_MANAGER_BATTERY_STATUS_UNKNOWN is 1

  constant ANDROID_OS_BATTERY_MANAGER_EXTRA_HEALTH is "health"
  constant ANDROID_OS_BATTERY_MANAGER_EXTRA_ICON_SMALL is "icon small"
  constant ANDROID_OS_BATTERY_MANAGER_EXTRA_LEVEL is "level"
  constant ANDROID_OS_BATTERY_MANAGER_EXTRA_PLUGGED is "plugged"
  constant ANDROID_OS_BATTERY_MANAGER_EXTRA_PRESENT is "present"
  constant ANDROID_OS_BATTERY_MANAGER_EXTRA_SCALE is "scale"
  constant ANDROID_OS_BATTERY_MANAGER_EXTRA_STATUS is "status"
  constant ANDROID_OS_BATTERY_MANAGER_EXTRA_TECHNOLOGY is "technology"
  constant ANDROID_OS_BATTERY_MANAGER_EXTRA_TEMPERATURE is "temperature"
  constant ANDROID_OS_BATTERY_MANAGER_EXTRA_VOLTAGE is "voltage"

  handler AndroidOsBatteryManagerGetIntProperty(in pId as Integer) \

```
      returns Integer
    wrap java int android.os.BatteryManager.getIntProperty(int id)
    in pId into id
    return result
  end handler

  handler AndroidOsBatteryManagerGetLongProperty(in pId as Integer) \
      returns Integer
    wrap java long android.os.BatteryManager.getLongProperty(int id)
    in pId into id
    return result
  end handler

  handler AndroidOsBatteryManagerIsCharging() \
      returns Boolean
    wrap java boolean android.os.BatteryManager.isCharging()
    return result
  end handler

end module
```