

# Handling the Unexpected

---

Unexpected Disconnection



**David Nutter**

IOS DEVELOPER

@NutterFi

[www.drumbeatninja.com](http://www.drumbeatninja.com)





Central



Peripheral



# Default Backgrounded Apps Can't Do This



**Peripheral - Advertise services**



**Peripheral - Provide updates to subscribed centrals**



**Central - Receive notify updates to subscribed characteristics**



# Scenario: Hard-to-Reach Peripheral



# Scenario: Central App Switching



# Handling the Unexpected

**Enable BLE background modes**

**Implement error handling**

**Use state preservation and restoration**



BLESensor > DX Finished running BLESensor on DX

BLESensor

AppDelegate.swift  
AppFlowController.swift  
AccelerometerViewController.swift  
AccelerometerViewController.xib  
BLEIdentifiers.swift  
BЛЕCentral.swift  
BLEPeripheral.swift  
AccelerometerDataSource.swift  
DeviceAccelerometerDataSource.swift  
BLERoleSelectViewController.swift  
PeripheralViewController.swift  
DiscoveredPeripheral.swift  
DiscoveredPeripheralCell.swift  
DiscoveredPeripheralCell.xib  
DiscoveryViewController.swift  
ViewController.swift  
Main.storyboard  
Assets.xcassets  
LaunchScreen.storyboard  
Info.plist

Products

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

PROJECT TARGETS

BLESensor

Access WiFi Information  
App Groups  
Apple Pay  
Associated Domains  
AutoFill Credential Provider  
Background Modes  
 ClassKit  
Data Protection  
Game Center  
HealthKit  
HomeKit  
Hotspot Configuration

BLESensor > DX Finished running BLESensor on DX

BLESensor

AppDelegate.swift  
AppFlowController.swift  
AccelerometerViewController.swift  
AccelerometerViewController.xib  
BLEIdentifiers.swift  
BЛЕCentral.swift  
BЛЕPeripheral.swift  
AccelerometerDataSource.swift  
DeviceAccelerometerDataSource.swift  
BLERoleSelectViewController.swift  
PeripheralViewController.swift  
DiscoveredPeripheral.swift  
DiscoveredPeripheralCell.swift  
DiscoveredPeripheralCell.xib  
DiscoveryViewController.swift  
ViewController.swift  
Main.storyboard  
Assets.xcassets  
LaunchScreen.storyboard  
Info.plist

Products

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

PROJECT TARGETS

BLESensor

BLESensor

Access WiFi Information OFF

App Groups OFF

Apple Pay OFF

Associated Domains OFF

AutoFill Credential Provider OFF

Background Modes ON

Modes:

- Audio, AirPlay, and Picture in Picture
- Location updates
- Voice over IP
- Newsstand downloads
- External accessory communication
- Uses Bluetooth LE accessories
- Acts as a Bluetooth LE accessory
- Background fetch
- Remote notifications

Steps: ✓ Add the Required Background Modes key to your info plist file

ClassKit OFF

Data Protection OFF

BLESensor > DX Finished running BLESensor on DX

BLESensor

AppDelegate.swift  
AppFlowController.swift  
AccelerometerViewController.swift  
AccelerometerViewController.xib  
BLEIdentifiers.swift  
BЛЕCentral.swift  
BЛЕPeripheral.swift  
AccelerometerDataSource.swift  
DeviceAccelerometerDataSource.swift  
BLERoleSelectViewController.swift  
PeripheralViewController.swift  
DiscoveredPeripheral.swift  
DiscoveredPeripheralCell.swift  
DiscoveredPeripheralCell.xib  
DiscoveryViewController.swift  
ViewController.swift  
Main.storyboard  
Assets.xcassets  
LaunchScreen.storyboard  
Info.plist

Products

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

PROJECT TARGETS

BLESensor

BLESensor

Access WiFi Information OFF

App Groups OFF

Apple Pay OFF

Associated Domains OFF

AutoFill Credential Provider OFF

Background Modes ON

Modes:

- Audio, AirPlay, and Picture in Picture
- Location updates
- Voice over IP
- Newsstand downloads
- External accessory communication
- Uses Bluetooth LE accessories
- Acts as a Bluetooth LE accessory
- Background fetch
- Remote notifications

Steps: ✓ Add the Required Background Modes key to your info plist file

ClassKit OFF

Data Protection OFF

BLESensor > DX Finished running BLESensor on DX

BLESensor

AppDelegate.swift  
AppFlowController.swift  
AccelerometerViewController.swift  
AccelerometerViewController.xib  
BLEIdentifiers.swift  
BLCentral.swift  
BLEPeripheral.swift  
AccelerometerDataSource.swift  
DeviceAccelerometerDataSource.swift  
BLERoleSelectViewController.swift  
PeripheralViewController.swift  
DiscoveredPeripheral.swift  
DiscoveredPeripheralCell.swift  
DiscoveredPeripheralCell.xib  
DiscoveryViewController.swift  
ViewController.swift  
Main.storyboard  
Assets.xcassets  
LaunchScreen.storyboard  
Info.plist

Products

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

PROJECT TARGETS

BLESensor

▼ Custom iOS Target Properties

Key	Type	Value
► Required device capabilities	Array	(1 item)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle versions string, short	String	1.0
Bundle version	String	1
► Required background modes	Array	(2 items)
Executable file	String	\$(EXECUTABLE_NAME)
Application requires iPhone environment	Boolean	YES
Launch screen interface file base name	String	LaunchScreen
► Supported interface orientations	Array	(3 items)
Bundle OS Type code	String	APPL
Localization native development region	String	\$(DEVELOPMENT_LANGUAGE)
► Supported interface orientations (iPad)	Array	(4 items)
Bundle name	String	\$(PRODUCT_NAME)

► Document Types (0)

► Exported UTIs (0)

► Imported UTIs (0)

► URL Types (0)

BLESensor > DX Finished running BLESensor on DX

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

PROJECT TARGETS

BLESensor

Custom iOS Target Properties

Key	Type	Value
► Required device capabilities	Array	(1 item)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle versions string, short	String	1.0
Bundle version	String	1
▼ Required background modes	Array	(2 items)
Item 0	String	App communicates using CoreBluetooth
Item 1	String	App shares data using CoreBluetooth
Executable file	String	\$(EXECUTABLE_NAME)
Application requires iPhone environment	Boolean	YES
Launch screen interface file base name	String	LaunchScreen
► Supported interface orientations	Array	(3 items)
Bundle OS Type code	String	APPL
Localization native development region	String	\$(DEVELOPMENT_LANGUAGE)
► Supported interface orientations (iPad)	Array	(4 items)
Bundle name	String	\$(PRODUCT_NAME)

► Document Types (0)

► Exported UTIs (0)

► Imported UTIs (0)

► URL Types (0)

+ Filter + - Filter

BLESensor > DX Finished running BLESensor on DX

BLESensor

AppDelegate.swift  
AppFlowController.swift  
AccelerometerViewController.swift  
AccelerometerViewController.xib  
BLEIdentifiers.swift  
BLCentral.swift  
BLEPeripheral.swift  
AccelerometerDataSource.swift  
DeviceAccelerometerDataSource.swift  
BLERoleSelectViewController.swift  
PeripheralViewController.swift  
DiscoveredPeripheral.swift  
DiscoveredPeripheralCell.swift  
DiscoveredPeripheralCell.xib  
DiscoveryViewController.swift  
ViewController.swift  
Main.storyboard  
Assets.xcassets  
LaunchScreen.storyboard  
Info.plist

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

PROJECT TARGETS

▼ Custom iOS Target Properties

Key	Type	Value
► Required device capabilities	Array	(1 item)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle versions string, short	String	1.0
Bundle version	String	1
▼ Required background modes	Array	(2 items)
Item 0	String	App communicates using CoreBluetooth
Item 1	String	App shares data using CoreBluetooth
Executable file	String	\$(EXECUTABLE_NAME)
Application requires iPhone environment	Boolean	YES
Launch screen interface file base name	String	LaunchScreen
► Supported interface orientations	Array	(3 items)
Bundle OS Type code	String	APPL
Localization native development region	String	\$(DEVELOPMENT_LANGUAGE)
► Supported interface orientations (iPad)	Array	(4 items)
Bundle name	String	\$(PRODUCT_NAME)
Privacy - Bluetooth Peripheral Usage Description	String	

► Document Types (0)  
► Exported UTIs (0)  
► Imported UTIs (0)  
► URL Types (0)

Privacy - Bluetooth Peripheral Usage Description  
Privacy - Calendars Usage Description  
Privacy - Camera Usage Description  
Privacy - Contacts Usage Description  
Privacy - Face ID Usage Description  
Privacy - Health Records Usage Description  
Privacy - Health Share Usage Description  
Privacy - Health Update Usage Description  
Privacy - HomeKit Usage Description  
Privacy - Location Always and When In Use Usage Description

BLESensor > DX Finished running BLESensor on DX

BLESensor

AppDelegate.swift  
AppFlowController.swift  
AccelerometerViewController.swift  
AccelerometerViewController.xib  
BLEIdentifiers.swift  
BЛЕCentral.swift  
BLEPeripheral.swift  
AccelerometerDataSource.swift  
DeviceAccelerometerDataSource.swift  
BLERoleSelectViewController.swift  
PeripheralViewController.swift  
DiscoveredPeripheral.swift  
DiscoveredPeripheralCell.swift  
DiscoveredPeripheralCell.xib  
DiscoveryViewController.swift  
ViewController.swift  
Main.storyboard  
Assets.xcassets  
LaunchScreen.storyboard  
Info.plist

Products

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

PROJECT TARGETS

Custom iOS Target Properties

Key	Type	Value
UIRequiredDeviceCapabilities	Array	(1 item)
CFBundleIdentifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
CFBundleInfoDictionaryVersion	String	6.0
CFBundleShortVersionString	String	1.0
CFBundleVersion	String	1
UIBackgroundModes	Array	(2 items)
Item 0	String	bluetooth-central
Item 1	String	bluetooth-peripheral
CFBundleExecutable	String	\$(EXECUTABLE_NAME)
LSRequiresiPhoneOS	Boolean	YES
UILaunchStoryboardName	String	LaunchScreen
UISupportedInterfaceOrientations	Array	(3 items)
CFBundlePackageType	String	APPL
CFBundleDevelopmentRegion	String	\$(DEVELOPMENT_LANGUAGE)
UISupportedInterfaceOrientations~ipad	Array	(4 items)
CFBundleName	String	\$(PRODUCT_NAME)
NSBluetoothPeripheralUsageDescription	String	

► Document Types (0)  
► Exported UTIs (0)  
► Imported UTIs (0)  
► URL Types (0)

+ Filter + - Filter

BLESensor > DX Finished running BLESensor on DX

BLESensor

AppDelegate.swift  
AppFlowController.swift  
AccelerometerViewController.swift  
AccelerometerViewController.xib  
BLEIdentifiers.swift  
BЛЕCentral.swift  
BLEPeripheral.swift  
AccelerometerDataSource.swift  
DeviceAccelerometerDataSource.swift  
BLERoleSelectViewController.swift  
PeripheralViewController.swift  
DiscoveredPeripheral.swift  
DiscoveredPeripheralCell.swift  
DiscoveredPeripheralCell.xib  
DiscoveryViewController.swift  
ViewController.swift  
Main.storyboard  
Assets.xcassets  
LaunchScreen.storyboard  
Info.plist

Products

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

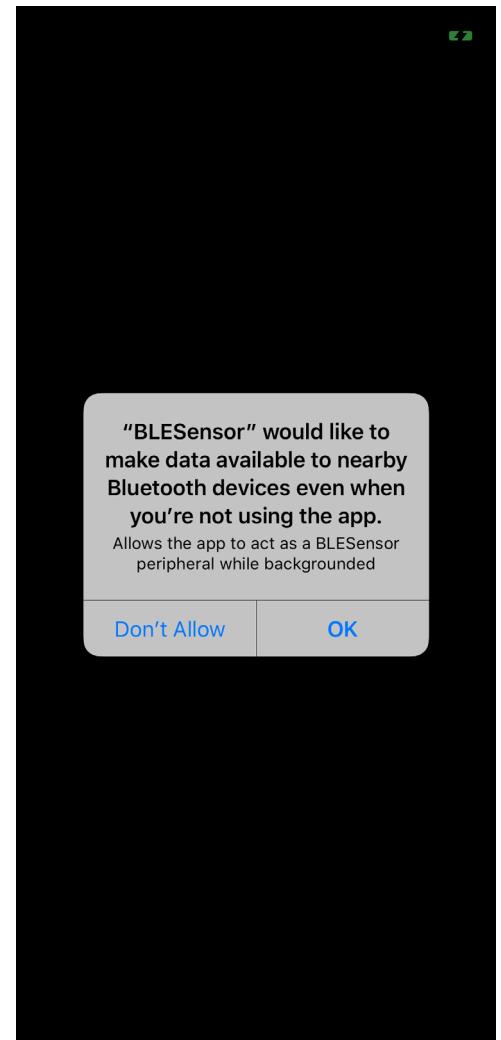
PROJECT TARGETS

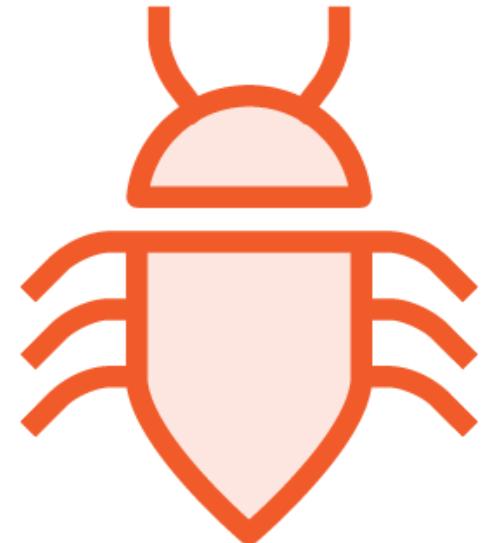
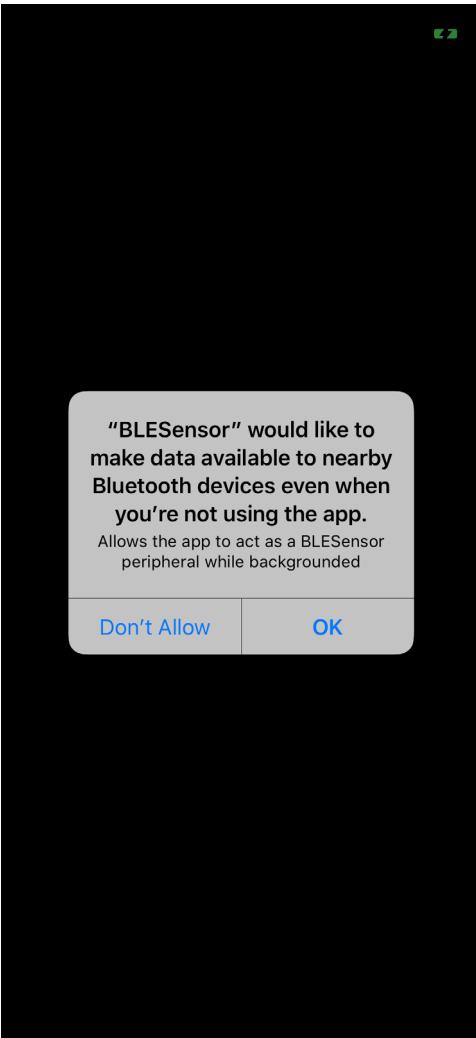
Custom iOS Target Properties

Key	Type	Value
UIRequiredDeviceCapabilities	Array	(1 item)
CFBundleIdentifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
CFBundleInfoDictionaryVersion	String	6.0
CFBundleShortVersionString	String	1.0
CFBundleVersion	String	1
UIBackgroundModes	Array	(2 items)
Item 0	String	bluetooth-central
Item 1	String	bluetooth-peripheral
CFBundleExecutable	String	\$(EXECUTABLE_NAME)
LSRequiresiPhoneOS	Boolean	YES
UILaunchStoryboardName	String	LaunchScreen
UISupportedInterfaceOrientations	Array	(3 items)
CFBundlePackageType	String	APPL
CFBundleDevelopmentRegion	String	\$(DEVELOPMENT_LANGUAGE)
UISupportedInterfaceOrientations~ipad	Array	(4 items)
CFBundleName	String	\$(PRODUCT_NAME)
NSBluetoothPeripheralUsageDescription	String	Allows the app to act as a BLESensor per

► Document Types (0)  
► Exported UTIs (0)  
► Imported UTIs (0)  
► URL Types (0)

+ Filter + - Filter





BLESensor &gt; DX

Running BLESensor on DX



**BLESensor PID 17422**

- CPU 9%
- Memory 13.8 MB
- Energy Impact High
- Disk 40 KB/s
- Network Zero KB/s

**Thread 1 Queue: com.apple.main-thread (serial)**

- 0 \_swift\_runtime\_on\_report
- 1 \_swift\_stdlib\_reportFatalError
- 2 specialized \_fatalErrorMessage(\_:\_:file:line:flags:)
- 3 \_fatalErrorMessage(\_:\_:file:line:flags:)
- 4 BLEPeripheral.update(with:)
- 5 closure #1 in BLEPeripheral.init(dataSource:)
- 6 closure #1 in DeviceAccelerometerDataSource.st...
- 7 thunk for @escaping @callee\_guaranteed (@guar...
- 8 \_\_NSBLOCKOPERATION\_IS\_CALLING\_OUT\_TO\_A\_...
- 9 -[NSBlockOperation main]
- 10 -[\_NSOperationInternal \_start:]
- 11 \_NSOQSchedule\_f
- 12 \_dispatch\_block\_async\_invoke2
- 13 \_dispatch\_client\_callout
- 14 \_dispatch\_main\_queue\_callback\_4CF
- 15 \_\_CFRUNLOOP\_IS\_SERVICING\_THE\_MAIN\_DISP...
- 16 \_\_CFRunLoopRun
- 17 CFRunLoopRunSpecific
- 18 GSEventRunModal
- 19 UIApplicationMain

```

40
41         service.characteristics = [characteristic]
42
43         manager.add(service)
44     }
45
46     func update(with data: AccelerometerData) {
47         if let payload = try? encoder.encode(data) {
48             characteristic.value = payload // Thread 1: Fatal error: Unexpectedly fo...
49             manager.updateValue(payload, for: characteristic,
50             onSubscribedCentrals: nil)
51         } else {
52             print("error encoding AccelerometerData")
53         }
54     }
55     // MARK: CBPeripheralManagerDelegate

```

Thread 1 < 4 BLEPeripheral.update(with:)

- A data (BLESensor.AccelerometerData)
- A self = (BLESensor.BLEPeripheral) 0x00000002837dc230
- ObjectiveC.NSObject (NSObject)
- manager = (CBPeripheralManager?) 0x0000000283fc83f0
- characteristic = (CBMutableCharacteristic?) nil
- encoder = (JSONEncoder) 0x00000002826c79f0
- dataSource (AccelerometerDataSource)
- L payload = (Data) 104 bytes

accelerometer data: x -0.010483 y -0.029129  
z -1.006058 @ 695096.755606  
Fatal error: Unexpectedly found nil while implicitly unwrapping an Optional value  
2019-07-11 01:00:35.092868-0600  
BLESensor[17422:5377383] Fatal error:  
Unexpectedly found nil while implicitly unwrapping an Optional value  
(lldb)

File □ Search □ Alert □ List □ Comment □ BLESensor > BLESensor > BLEPeripheral.swift > BLEPeripheral

```
55 // MARK: CBPeripheralManagerDelegate
56
57 func peripheralManagerDidUpdateState(_ peripheral: CBPeripheralManager) {
58     if peripheral.state == .poweredOn {
59         setup()
60     } else {
61         print("peripheral is not available: \(peripheral.state.rawValue)")
62     }
63 }
64
65 func peripheralManager(_ peripheral: CBPeripheralManager, didAdd service:
66 CBService, error: Error?) {
67     if let error = error {
68         print("Could not add service: \(error.localizedDescription)")
69     } else {
70         print("peripheral added service. Start advertising")
71         let advertisementData: [String: Any] = [
72             CBAdvertisementDataServiceUUIDsKey: [CBUUID(string:
73                 BLEIdentifiers.serviceIdentifier)],
74             CBAdvertisementDataLocalNameKey: "BLE Sensor" // This key will not be
75                                         // transmitted when app is backgrounded
76         ]
77         manager.startAdvertising(advertisementData)
78     }
79 }
80 }
```

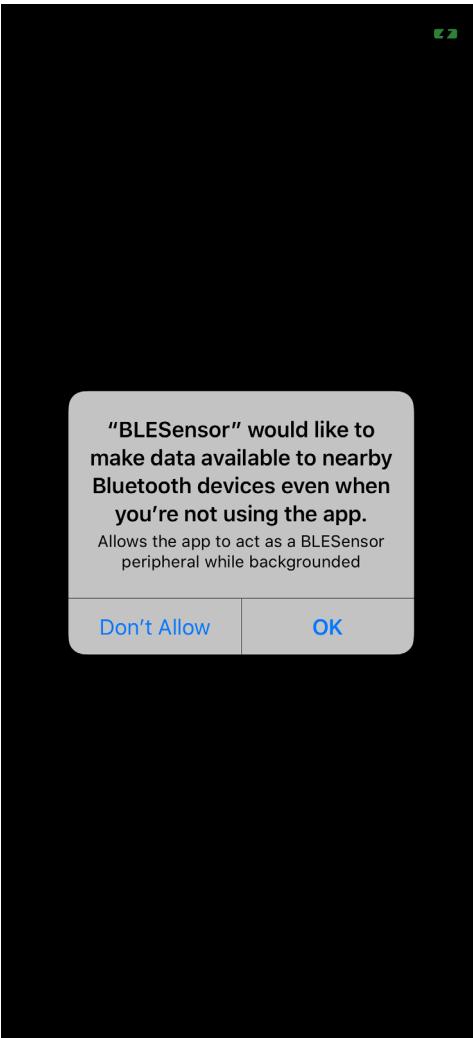
The screenshot shows the Xcode interface with the following details:

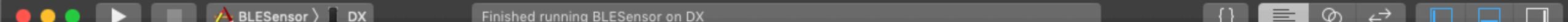
- Top Bar:** Xcode, File, Edit, View, Find, Navigate, Editor, Product, Debug, Source Control, Window, Help.
- Toolbar:** Standard Xcode toolbar with icons for play/pause, stop, run, and file operations.
- Project Navigator:** Shows the project structure under "BLESensor".
- Search Bar:** Finished running BLESensor on DX.
- Editor:** Displays the code for `BLEPeripheral.swift`. The code is as follows:

```
15     private var characteristic: CBMutableCharacteristic!
16     private let encoder = JSONEncoder()
17
18     private var dataSource: AccelerometerDataSource
19
20     init(dataSource: AccelerometerDataSource) {
21         self.dataSource = dataSource
22         super.init()
23         manager = CBPeripheralManager(delegate: self, queue: nil)
24         self.dataSource.onUpdate = { [weak self] data in
25             self?.update(with: data)
26         }
27         self.dataSource.start()
28     }
29
30     func setup() {
31         let characteristicUUID = CBUUID(string:
32             BLEIdentifiers.characteristicIdentifier)
33
34         characteristic = CBMutableCharacteristic(type: characteristicUUID, properties:
35             [.read, .notify], value: nil, permissions: [.readable])
36
37         let descriptor = CBMutableDescriptor(type: CBUUID(string:
38             CBUUIDCharacteristicUserDescriptionString), value: "BLESensor prototype")
39         characteristic.descriptors = [descriptor]
```



```
40
41     service.characteristics = [characteristic]
42
43     manager.add(service)
44 }
45
46 func update(with data: AccelerometerData) {
47     if let payload = try? encoder.encode(data), characteristic != nil {
48         characteristic.value = payload
49         manager.updateValue(payload, for: characteristic, onSubscribedCentrals:
50                             nil)
51     } else {
52         print("error encoding AccelerometerData")
53     }
54 }
55 // MARK: CBPeripheralManagerDelegate
56
57 func peripheralManagerDidUpdateState(_ peripheral: CBPeripheralManager) {
58     if peripheral.state == .poweredOn {
59         setup()
60     } else {
61         print("peripheral is not available: \(peripheral.state.rawValue)")
62     }
63 }
64
```





BLESensor > DX Finished running BLESensor on DX

File View Insert Editor Run Product Scheme Build Window Help

BLESensor

BLESensor

AppDelegate.swift

AppFlowController.swift

AccelerometerController.swift

AccelerometerController.xib

BLEIdentifiers.swift

BLECentral.swift

BLEPeripheral.swift

AccelerometerSource.swift

DeviceAccel...aSource.swift

BLERoleSele...ontroller.swift

PeripheralVi...ontroller.swift

DiscoveredPeripheral.swift

DiscoveredP...eralCell.swift

DiscoveredP...ipheralCell.xib

DiscoveryVie...ntroller.swift

ViewController.swift

Main.storyboard

Assets.xcassets

LaunchScreen.storyboard

Info.plist

Products

40                  service.characteristics = [characteristic]  
41  
42                  manager.add(service)  
43      }  
44  
45  
46      func update(with data: AccelerometerData) {  
47          if let payload = try? encoder.encode(data), characteristic != nil {  
48              characteristic.value = payload  
49              manager.updateValue(payload, for: characteristic, onSubscribedCentrals:  
50                  nil)  
51          } else {  
52              print("error encoding AccelerometerData")  
53          }  
54      }  
55      // MARK: CBPeripheralManagerDelegate

accelerometer data: x 0.010330 y 0.058319 z -0.993195 @ 725246.440286  
error encoding AccelerometerData  
peripheral added service. Start advertising  
peripheral started advertising  
accelerometer data: x -0.003006 y 0.051132 z -1.001160 @ 725247.440002  
accelerometer data: x -0.032898 y 0.069351 z -0.997452 @ 725248.439686  
accelerometer data: x -0.036560 y 0.043671 z -0.999832 @ 725249.439351  
accelerometer data: x -0.027588 y 0.058014 z -1.001831 @ 725250.439044  
accelerometer data: x -0.005356 y 0.050125 z -1.008423 @ 725251.438704  
accelerometer data: x 0.024682 y 0.052101 z -1.005240 @ 725252.439245

# Error Handling

---



# Delegate Methods Contain Error Parameters

**CBCentralManagerDelegate**

**CBPeripheralDelegate**

**CBPeripheralManagerDelegate**



## Monitoring Connections with Peripherals

```
func centralManager(CBCentralManager, didConnect: CBPeripheral)
    Invoked when a connection is successfully created with a peripheral.
```

```
func centralManager(CBCentralManager, didDisconnectPeripheral: CBPeripheral, error: Error?)
    Invoked when an existing connection with a peripheral is torn down.
```

```
func centralManager(CBCentralManager, didFailToConnect: CBPeripheral, error: Error?)
    Invoked when the central manager fails to create a connection with a peripheral.
```

## Discovering and Retrieving Peripherals

```
func centralManager(CBCentralManager, didDiscover: CBPeripheral, advertisementData: [String : Any], rssi: NSNumber)
    Invoked when the central manager discovers a peripheral while scanning.
```

## Monitoring Changes to the Central Manager's State

```
func centralManagerDidUpdateState(CBCentralManager)
    Invoked when the central manager's state is updated.
    Required.
```

```
func centralManager(CBCentralManager, willRestoreState: [String : Any])
    Invoked when the central manager is about to be restored by the system.
```

## Discovering Services

```
func peripheral(CBPeripheral, didDiscoverServices: Error?)  
    Invoked when you discover the peripheral's available services.  
  
func peripheral(CBPeripheral, didDiscoverIncludedServicesFor: CBService, error: Error?)  
    Invoked when you discover the included services of a specified service.
```

## Discovering Characteristics and Characteristic Descriptors

```
func peripheral(CBPeripheral, didDiscoverCharacteristicsFor: CBService, error: Error?)  
    Invoked when you discover the characteristics of a specified service.  
  
func peripheral(CBPeripheral, didDiscoverDescriptorsFor: CBCharacteristic, error: Error?)  
    Invoked when you discover the descriptors of a specified characteristic.
```

## Retrieving Characteristic and Characteristic Descriptor Values

```
func peripheral(CBPeripheral, didUpdateValueFor: CBCharacteristic, error: Error?)  
    Invoked when you retrieve a specified characteristic's value, or when the peripheral device notifies your app  
    that the characteristic's value has changed.  
  
func peripheral(CBPeripheral, didUpdateValueFor: CBDescriptor, error: Error?)  
    Invoked when you retrieve a specified characteristic descriptor's value.
```

```
protocol CBPeripheralManagerDelegate
```

Declaration  
Topics  
Relationships

## Topics

---

### Monitoring Changes to the Peripheral Manager's State

```
func peripheralManagerDidUpdateState(CBPeripheralManager)
```

Invoked when the peripheral manager's state is updated.

Required.

```
func peripheralManager(CBPeripheralManager, willRestoreState: [String : Any])
```

Invoked when the peripheral manager is about to be restored by the system.

### Adding Services

---

```
func peripheralManager(CBPeripheralManager, didAdd: CBService, error: Error?)
```

Invoked when you publish a service, and any of its associated characteristics and characteristic descriptors, to the local Generic Attribute Profile (GATT) database.

### Advertising Peripheral Data

---

```
func peripheralManagerDidStartAdvertising(CBPeripheralManager, error: Error?)
```

Invoked when you start advertising the local peripheral device's data.

# Error Types

CBError.Code

**The possible errors returned during Bluetooth low energy transactions**

CBATTError.Code

**The possible errors returned by a GATT server (a remote peripheral) during Bluetooth low energy ATT transactions.**



# Central Related Errors

**State changes in central manager**

**Peripheral connections**

**Peripheral discovery of services, characteristics, descriptors**

**Reading, writing, notify updates to characteristic and descriptor values**



```
func centralManager(_ central: CBCentralManager, didFailToConnect  
peripheral: CBPeripheral, error: Error?) {  
    central.connect(peripheral) // try again  
}
```

## centralManager(\_:didFailToConnect:error:)

**Invoked when a connection initiated via connect(\_:options:) fails to complete.**

**Connection attempts do not time out, so this method usually indicates a transient issue. Can simply reattempt to connect to the peripheral.**



# Unexpected Disconnection from Peripheral

**Centrals can initiate disconnection with  
`cancelPeripheralConnection()`**

**On disconnection,  
`centralManager(_:didDisconnectPeripheral:  
error:)` is invoked**

**All peripheral services, characteristics and  
descriptors become invalid**

**Error parameter only present if  
disconnection is unintentional**



# Unexpected Disconnection from Peripheral

```
func centralManager(_ central: CBCentralManager,  
didDisconnectPeripheral peripheral: CBPeripheral, error: Error?) {  
  
    if let error = error {  
        // Handle disconnection error  
    } else {  
        print("Nothing to see here...")  
    }  
}
```



# Handling State Changes

```
public enum CBManagerState : Int {  
    case unknown  
    case resetting  
    case unsupported  
    case unauthorized  
    case poweredOff  
    case poweredOn  
}
```



# Handling State Changes

**When state is `.poweredOff`, all scanning stops and connections to peripherals have ended**

**When state is neither `.poweredOn` nor `.poweredOff`, all `CBPeripheral` objects are invalid**

**Peripherals must be retrieved or discovered again**



# State Change Scenario



**.poweredOn** - connected to remote peripheral



**.poweredOff** – centralManagerDidUpdateState(\_:) and  
centralManager(\_:didDisconnectPeripheral:error:) called



**.poweredOn** – centralManagerDidUpdateState(\_:) called



# Handling State Changes Example - Old Way

```
func centralManagerDidUpdateState(_ central: CBCentralManager) {  
    switch central.state {  
        case .poweredOn:  
            scanForPeripherals() // not necessary if we've connected before  
        // handle other cases...  
    }  
}
```



# Handling State Changes Example - New Way

```
func centralManagerDidUpdateState(_ central: CBCentralManager) {  
    switch central.state {  
        case .poweredOn:  
            if let lastConnectedPeripheral = lastConnectedPeripheral {  
                central.connect(lastConnectedPeripheral, options: nil)  
            } else {  
                scanForPeripherals()  
            }  
        // handle other cases...  
    }  
}
```

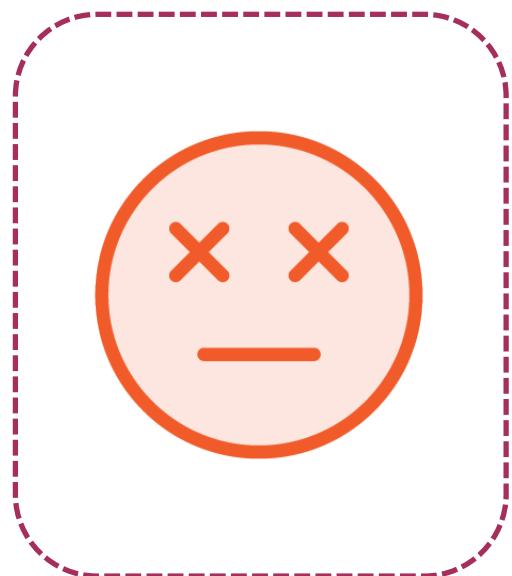


# State Preservation and Restoration

---



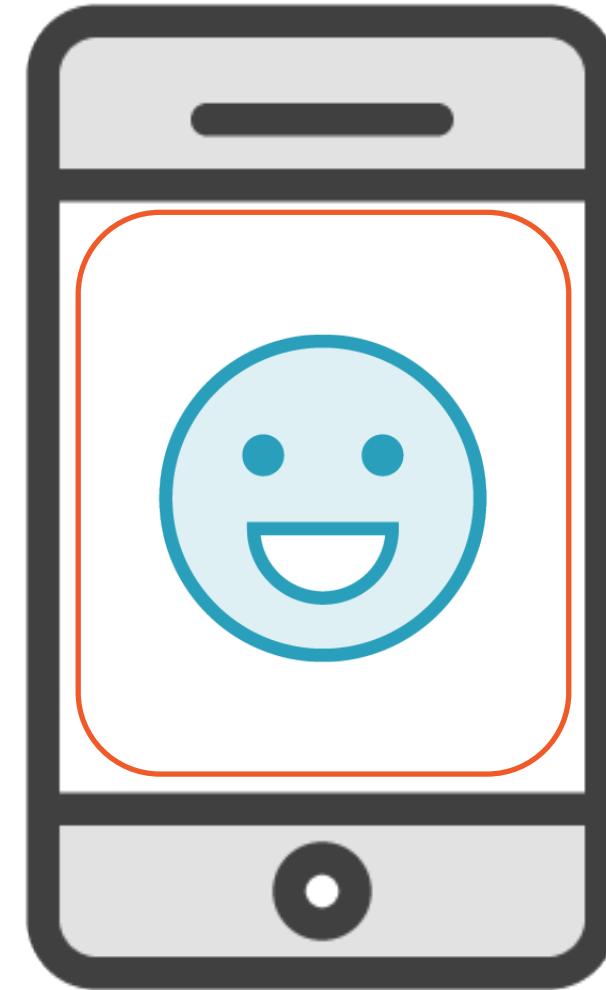
# Backgrounded Apps Can Be Terminated



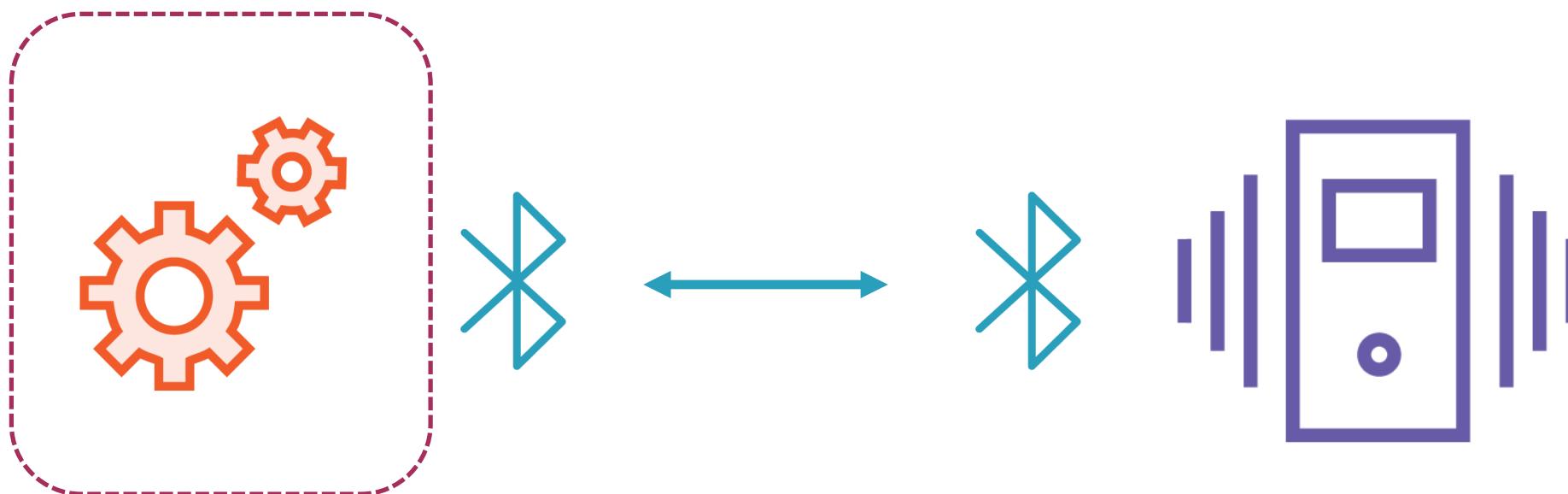
Terminated  
Backgrounded  
App



Backgrounded  
App



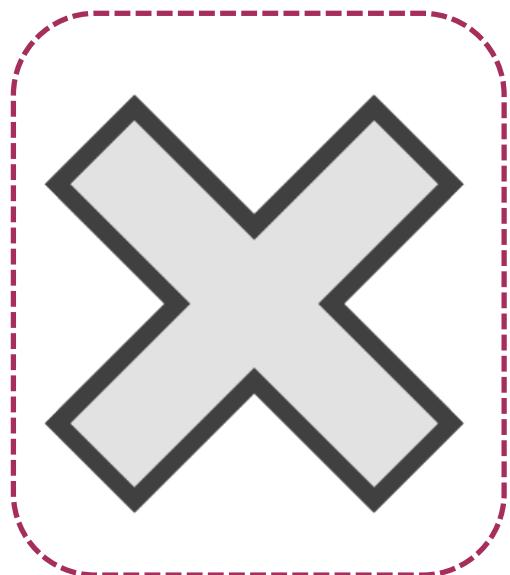
# Backgrounded BLE Communication



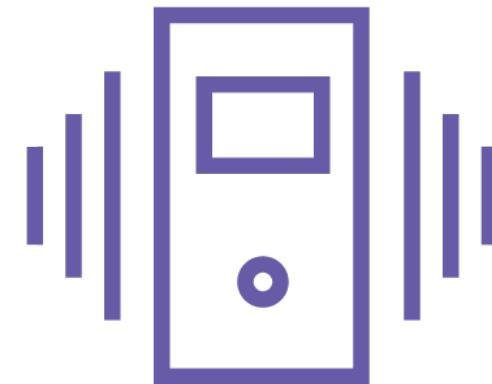
Backgrounded  
App



# Backgrounded BLE Communication



Terminated  
Backgrounded  
App



# State Preservation and Restoration

Core Bluetooth supported - allows system to preserve state of central and peripheral managers on app termination, and restoring that state at app launch time



# State Preservation and Restoration



**State preservation can survive device restarts**



**State preservation can survive Bluetooth related system events**



**System restarts your app in background when it detects Bluetooth event related to your app**



# State Preservation Bluetooth Tracking

## Centrals

Scanning for services, including any specified scan options

Peripherals the central manager was connected to or trying to connect to

Subscribed characteristics

## Peripherals

Data the peripheral was advertising

Services and characteristics that were published to the device's database

Centrals that were subscribed to characteristics values



# Adding Support for State Preservation and Restoration

- Opt in when initializing manager objects**
- Reinstantiate manager objects after app is relaunched by the system**
- Implement appropriate restoration delegate method**
- Update initialization process for managers as appropriate**



# Opt In to State Preservation and Restoration

```
let identifier = "myCentralIdentifier"

let options: [String : Any] =
[CBCentralManagerOptionRestoreIdentifierKey: identifier]

let manager = CBCentralManager(delegate: self, queue: nil, options:
options)

// use CBPeripheralManagerOptionRestoreIdentifierKey for peripherals
```



# Reinstantiate Managers After App Launch

```
func application(_ application: UIApplication,  
didFinishLaunchingWithOptions launchOptions:  
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {  
  
    let centralIdentifiers =  
    launchOptions?[UIApplication.LaunchOptionsKey.bluetoothCentrals] as?  
    [String]  
  
    // UIApplication.LaunchOptionsKey.bluetoothPeripherals  
  
    if let identifiers = centralIdentifiers,  
    identifiers.contains("myCentralIdentifier") {  
  
        buildCentral()  
  
    }  
}
```



# Restore Managers By Syncing Bluetooth State

```
func centralManager(_ central: CBCentralManager, willRestoreState  
dict: [String : Any]) {  
  
    // check dictionary for central state restoration options  
  
}  
  
func peripheralManager(_ peripheral: CBPeripheralManager,  
willRestoreState dict: [String : Any]) {  
  
    // check dictionary for peripheral state restoration options  
  
}  
  
// These methods are called BEFORE didUpdateState() methods
```



# Central Manager State Restoration Options

Keys used to pass options to the `init(delegate:queue:options:)` method.

Language  
Swift | Objective-C

Framework  
Core Bluetooth

## Topics

---

### Constants

`let CBCentralManagerRestoredStatePeripheralsKey: String`

An array (an instance of `NSArray`) of `CBPeripheral` objects that contains all of the peripherals that were connected to the central manager (or had a connection pending) at the time the app was terminated by the system.

`let CBCentralManagerRestoredStateScanServicesKey: String`

An array (an instance of `NSArray`) of service UUIDs (represented by `CBUUID` objects) that contains all the services the central manager was scanning for at the time the app was terminated by the system.

`let CBCentralManagerRestoredStateScanOptionsKey: String`

A dictionary (an instance of `NSDictionary`) that contains all of the peripheral scan options that were being used by the central manager at the time the app was terminated by the system.



# CBCentralManagerRestoredStatePeripheralsKey

An array (an instance of [NSArray](#)) of [CBPeripheral](#) objects that contains all of the peripherals that were connected to the central manager (or had a connection pending) at the time the app was terminated by the system.

Language

Swift

Objective-C

SDKs

iOS 7.0+

macOS 10.13+

tvOS 9.0+

watchOS 2.0+

Framework

Core Bluetooth

On This Page

[Declaration](#)

[Discussion](#)

## Declaration

```
let CBCentralManagerRestoredStatePeripheralsKey: String
```

## Discussion

When possible, all the information about a peripheral is restored, including any discovered services, characteristics, characteristic descriptors, and characteristic



Global Variable

# CBCentralManagerRestoredStateScanServicesKey

An array (an instance of [NSArray](#)) of service UUIDs (represented by [CBUUID](#) objects) that contains all the services the central manager was scanning for at the time the app was terminated by the system.

Language

Swift

Objective-C

SDKs

iOS 7.0+

macOS 10.13+

tvOS 9.0+

watchOS 2.0+

Framework

Core Bluetooth

```
let CBCentralManagerRestoredStateScanServicesKey: String
```



Global Variable

# CBCentralManagerRestoredStateScanOptionsKey

A dictionary (an instance of [NSDictionary](#)) that contains all of the peripheral scan options that were being used by the central manager at the time the app was terminated by the system.

Language

Swift

Objective-C

SDKs

iOS 7.0+

macOS 10.13+

tvOS 9.0+

watchOS 2.0+

Framework

Core Bluetooth

## Declaration

```
let CBCentralManagerRestoredStateScanOptionsKey: String
```



# Peripheral Manager State Restoration Options

Keys used to pass options to the `init(delegate:queue:options:)` method.

Language  
Swift | Objective-C

Framework  
Core Bluetooth

## Topics

---

### Constants

`let CBPeripheralManagerRestoredStateServicesKey: String`

An array (an instance of `NSArray`) of `CBMutableService` objects that contains all of the services that were published to the local peripheral's database at the time the app was terminated by the system.

`let CBPeripheralManagerRestoredStateAdvertisementDataKey: String`

A dictionary (an instance of `NSDictionary`) containing the data that the peripheral manager was advertising at the time the app was terminated by the system.



Global Variable

# CBPeripheralManagerRestoredStateServicesKey

An array (an instance of [NSArray](#)) of [CBMutableService](#) objects that contains all of the services that were published to the local peripheral's database at the time the app was terminated by the system.

Language

Swift

Objective-C

SDKs

iOS 7.0+

macOS 10.9+

tvOS 9.0+

watchOS 2.0+

Framework

Core Bluetooth

On This Page

[Declaration](#)

[Discussion](#)

## Declaration

```
let CBPeripheralManagerRestoredStateServicesKey: String
```

## Discussion

All the information about a service is restored, including any included services, characteristics, characteristic descriptors, and subscribed centrals.



Global Variable

# CBPeripheralManagerRestoredStateAdvertisementDataKey

A dictionary (an instance of [NSDictionary](#)) containing the data that the peripheral manager was advertising at the time the app was terminated by the system.

Language

Swift

Objective-C

SDKs

iOS 7.0+

macOS 10.9+

tvOS 9.0+

watchOS 2.0+

Framework

Core Bluetooth

## Declaration

```
let CBPeripheralManagerRestoredStateAdvertisementDataKey: :
```



# Central Restored State Example

```
func centralManager(_ central: CBCentralManager, willRestoreState  
dict: [String : Any]) {  
  
    if let peripherals =  
dict[CBCentralManagerRestoredStatePeripheralsKey] as? [CBPeripheral] {  
  
        self.restoredPeripherals = peripherals  
  
        // manage restored peripherals when state is powered on  
  
    }  
  
}
```



# Peripheral Restored State Example

```
func peripheralManager(_ peripheral: CBPeripheralManager,  
willRestoreState dict: [String : Any]) {  
  
    if let services =  
dict[CBPeripheralManagerRestoredStateServicesKey] as?  
[CBMutableService] {  
  
        self.restoredServices = services  
  
        // manage restored services when state is powered on  
  
    }  
  
}
```



# Restored Peripheral Service Discovery

```
restoredPeripherals.forEach { (peripheral) in
    peripheral.delegate = self
    if let serviceIdentifiers =
        peripheral.services?.compactMap({$0.uuid.uuidString}),
        serviceIdentifiers.contains(BLEIdentifiers.serviceIdentifier) {
        // service was already discovered – check characteristics...
    } else {
        // need to discover services
    }
}
```



# Summary



## Enable background modes for BLE communication

- Peripherals advertise services and notify subscribed centrals of characteristic updates
- Centrals receive characteristic updates from peripherals

## Handle errors for better user experience

- Error parameter in most delegate methods provides context
- Some errors are recoverable (e.g. peripheral connection, read characteristic)
- Others require canceling connection and retrying

## Use state preservation and restoration

- Central and peripheral state can be preserved after app termination
- Helpful for long-running tasks

