```swift
//
//  EmittingViewController.swift
//  MacBeacon
//

import Cocoa
import CoreBluetooth

class EmittingViewController: NSViewController,  CBPeripheralManagerDelegate  {

    // MARK: IBOutlets

    @IBOutlet weak var roomNumberLabel: NSTextField!

    // MARK: Public properties
    let uuid:NSUUID = NSUUID(uuidString: "DCEF54A2-31EB-467F-AF8E-350FB641C97B")!
    var peripheralManager:CBPeripheralManager = CBPeripheralManager()
    var beacon: Beacon?

    // MARK: NSViewController methods

    override func viewDidLoad() {
        super.viewDidLoad()

        roomNumberLabel.stringValue = (beacon?.roomNumber)!

        peripheralManager = CBPeripheralManager(delegate: self, queue: nil)
    }

    // MARK: IBAction methods

    @IBAction func stopTransmitting(_ sender: Any) {
        // Stop emitting and exit the view
        peripheralManager.stopAdvertising()
        dismiss(nil)
    }

    // MARK: Internal methods

    internal func peripheralManagerDidUpdateState(_ peripheral: CBPeripheralManager)
     {
        print("\(peripheral.description)")
        // Check the state of the peripherla manager: for debugging purposes
        switch peripheral.state {
            case .poweredOff:
                print("Powered off")
            case .poweredOn:
                print("Powered on")
            case .resetting:
                print("Resetting")
            case .unauthorized:
                print("Unauthorized")
            case .unknown:
                print("Unknown")
```

```
            case .unsupported:
                print("Unsupported")
        }
        startTransmitting()
    }

    internal func startTransmitting() {
        // Create the beacon packet with the library function
        let beaconPacket = CBBeaconAvertisementData(proximityUUID: uuid, major:
         UInt16((beacon?.major!)!), minor:  UInt16((beacon?.minor!)!),
         measuredPower: Int8(-60))
        if let advertisement = beaconPacket.beaconAdvertisement() {
            // Give the peripheral manager the packet and start advertising the
             packet
            peripheralManager.startAdvertising(advertisement as? [String : Any])
        }
    }
}
```