Яндекс Go

Как Яндекс Самокаты BLE внедряли

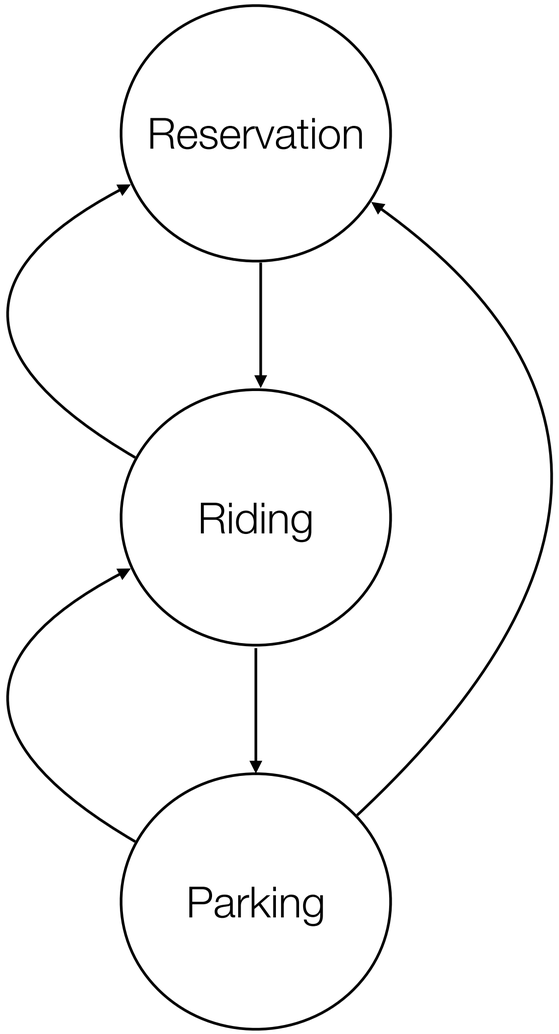
Кривозубов Влад, разработчик

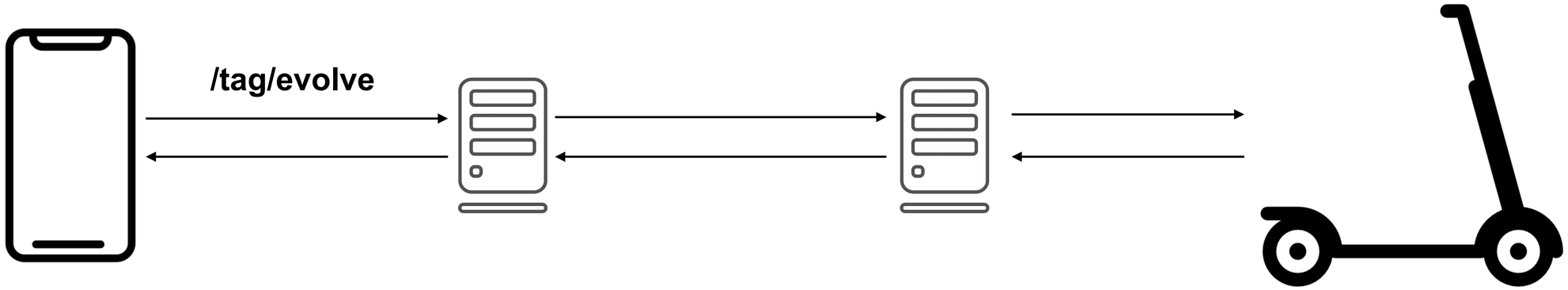
Содержание

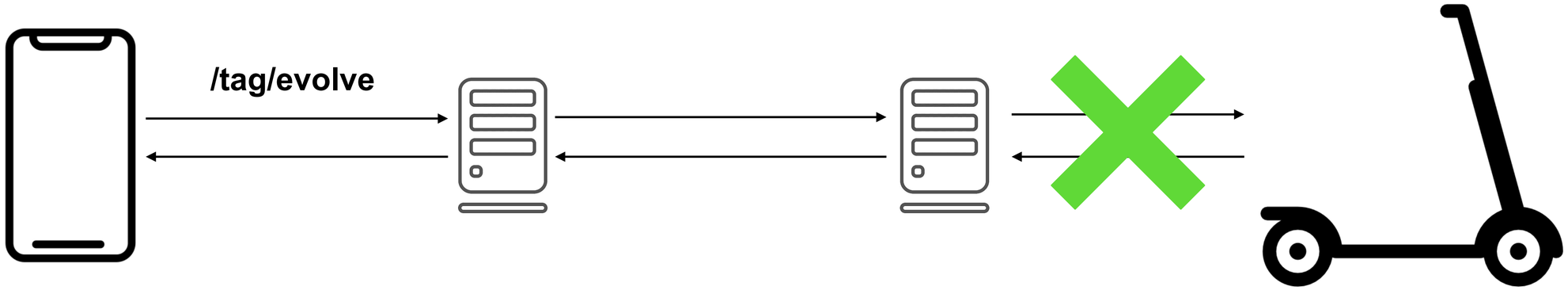
Содержание

Цикл заказа

/tag/evolve endpoint



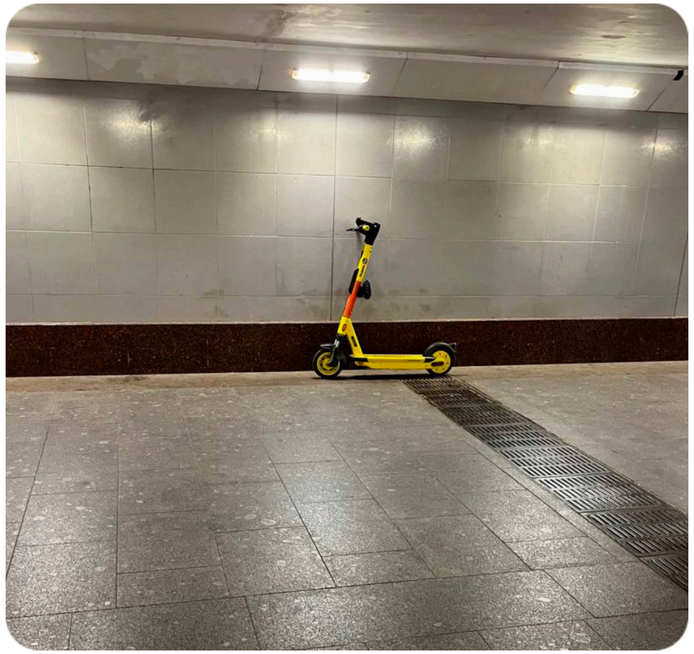

/tag/evolve
Продуктовый бекэнд
Бекэнд самокатов


/tag/evolve
Продуктовый бекэнд
Бекэнд самокатов

Где самокаты?

1 Замена оператора

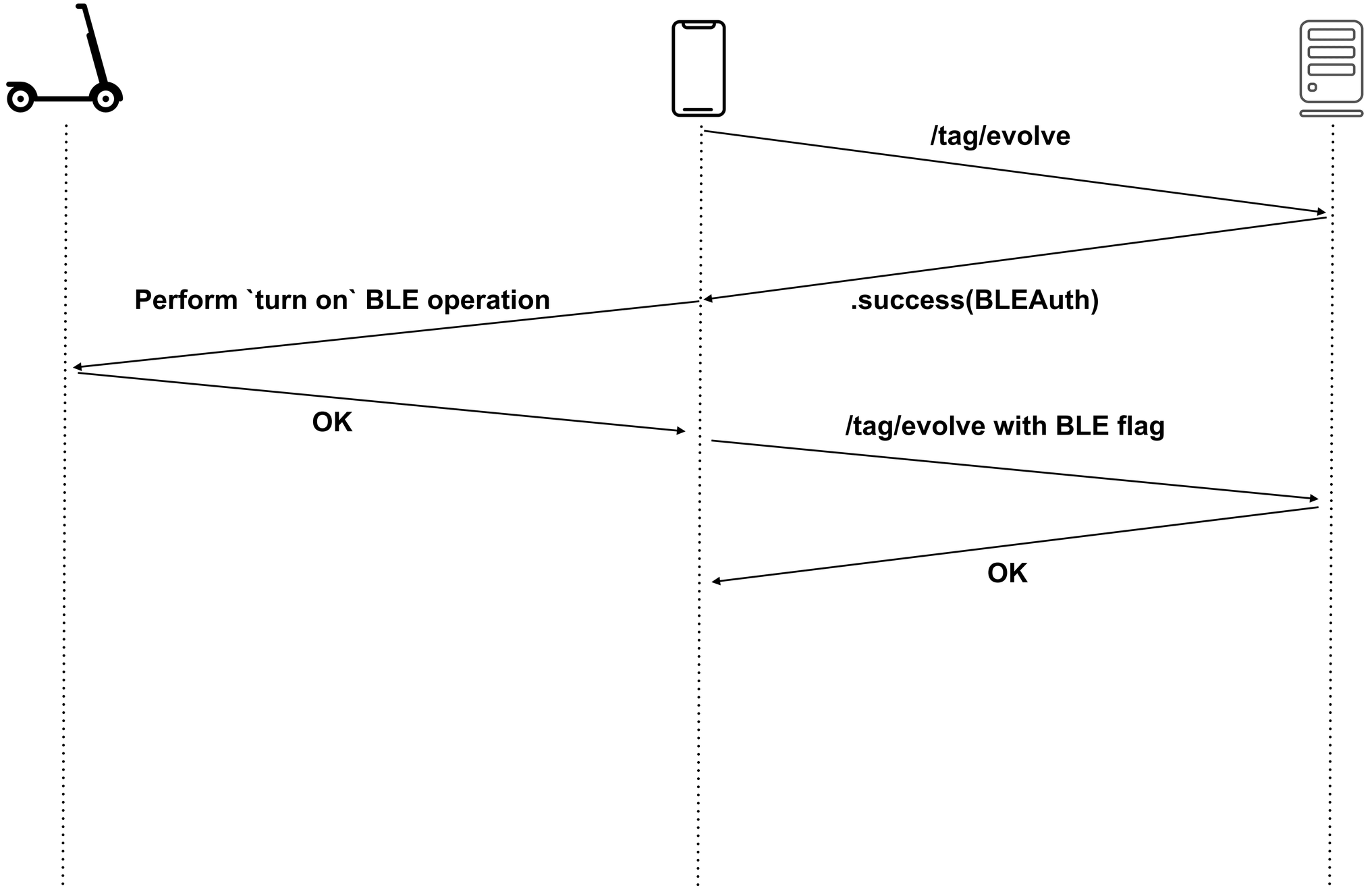
2 Использовать bluetooth на клиенте

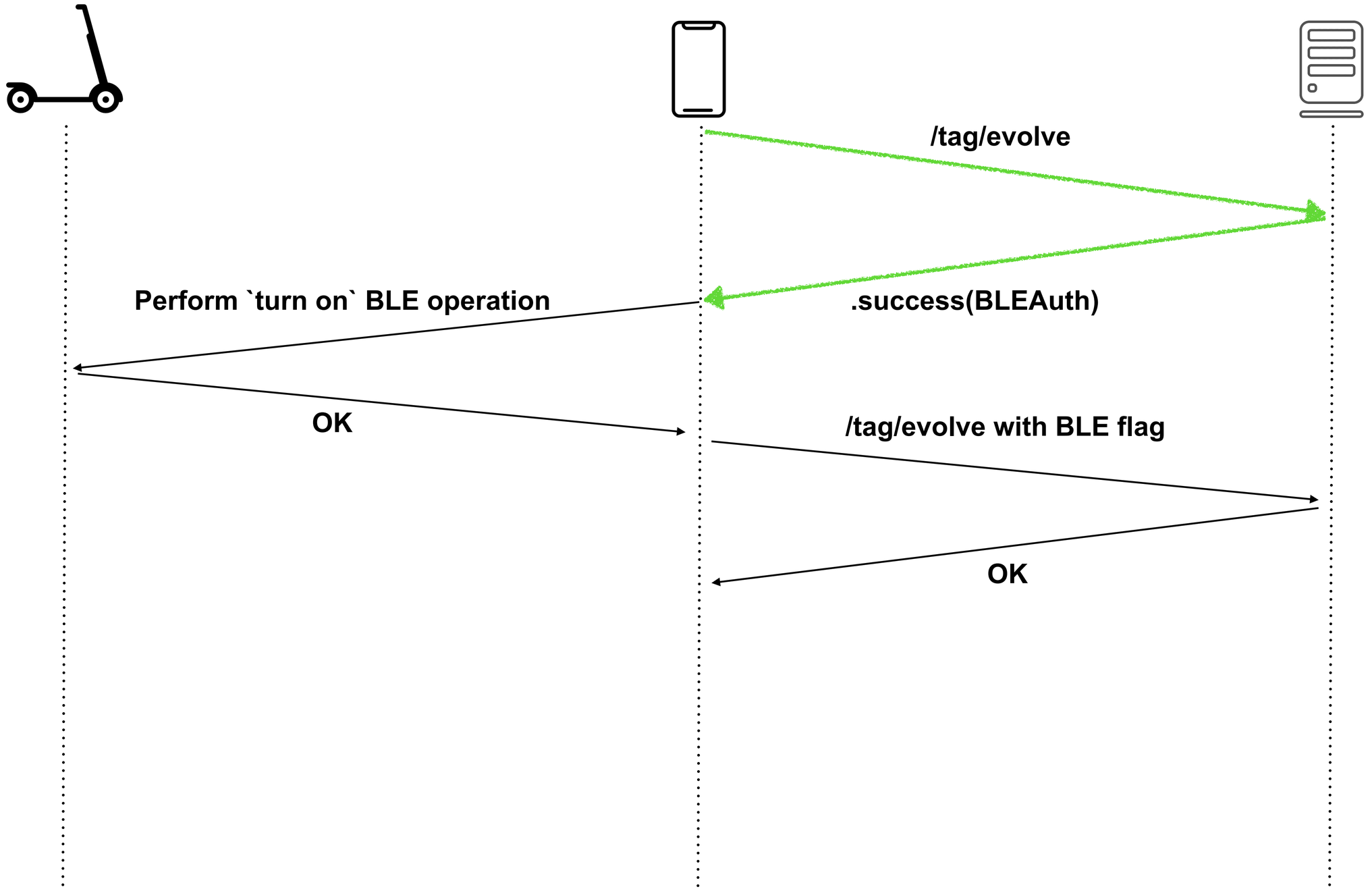


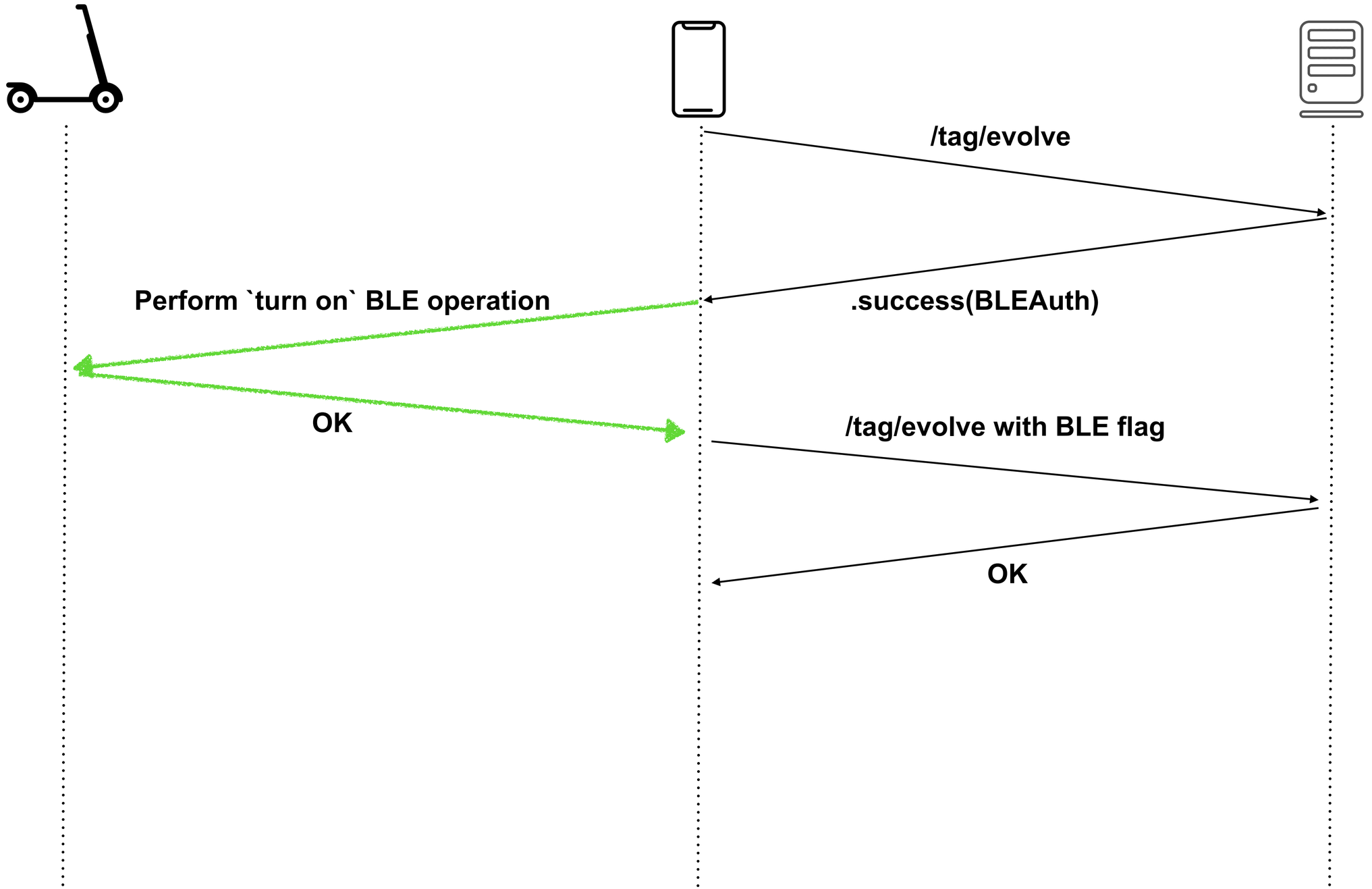
Содержание

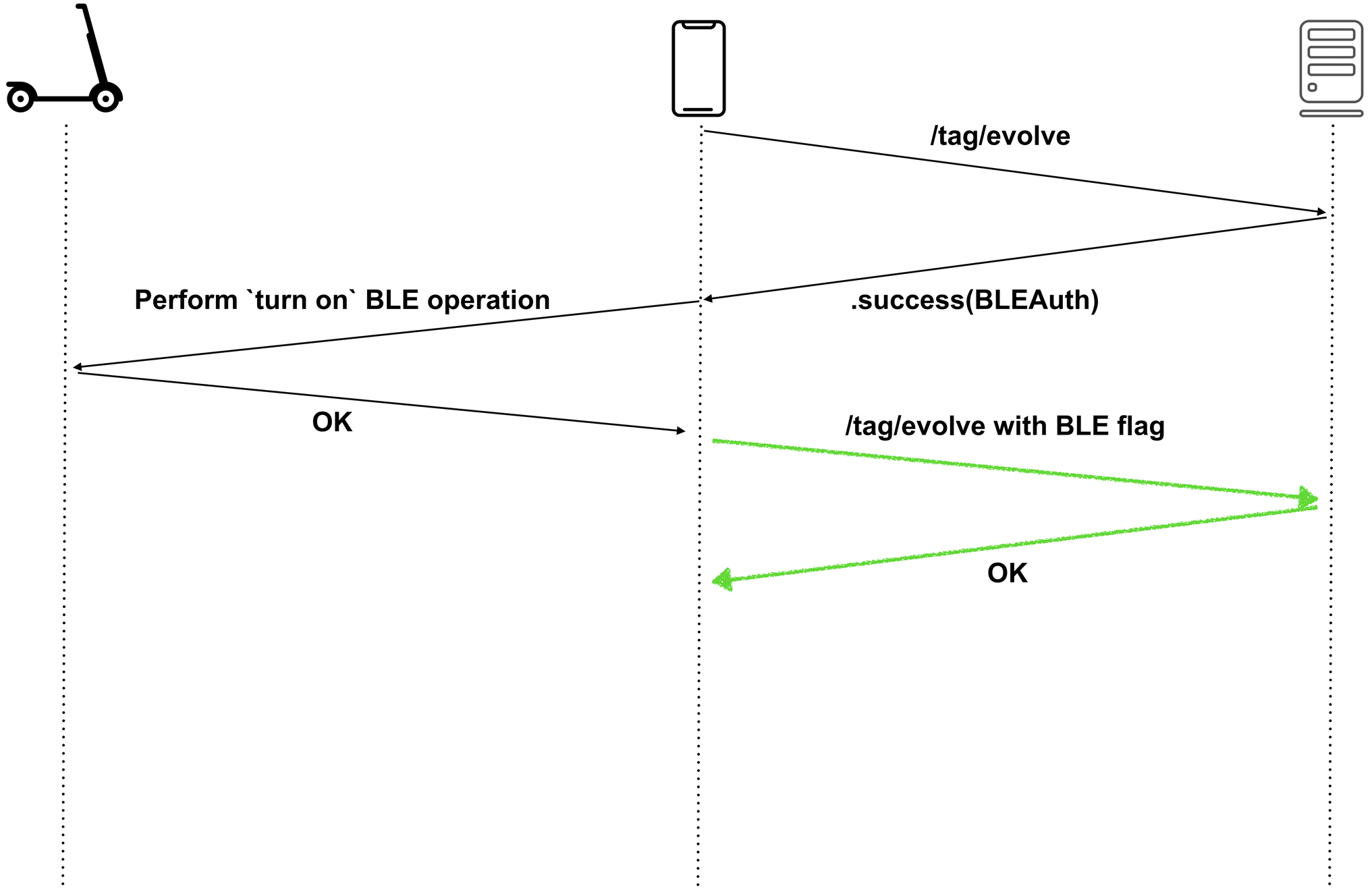
1 Получить данные нужного самоката

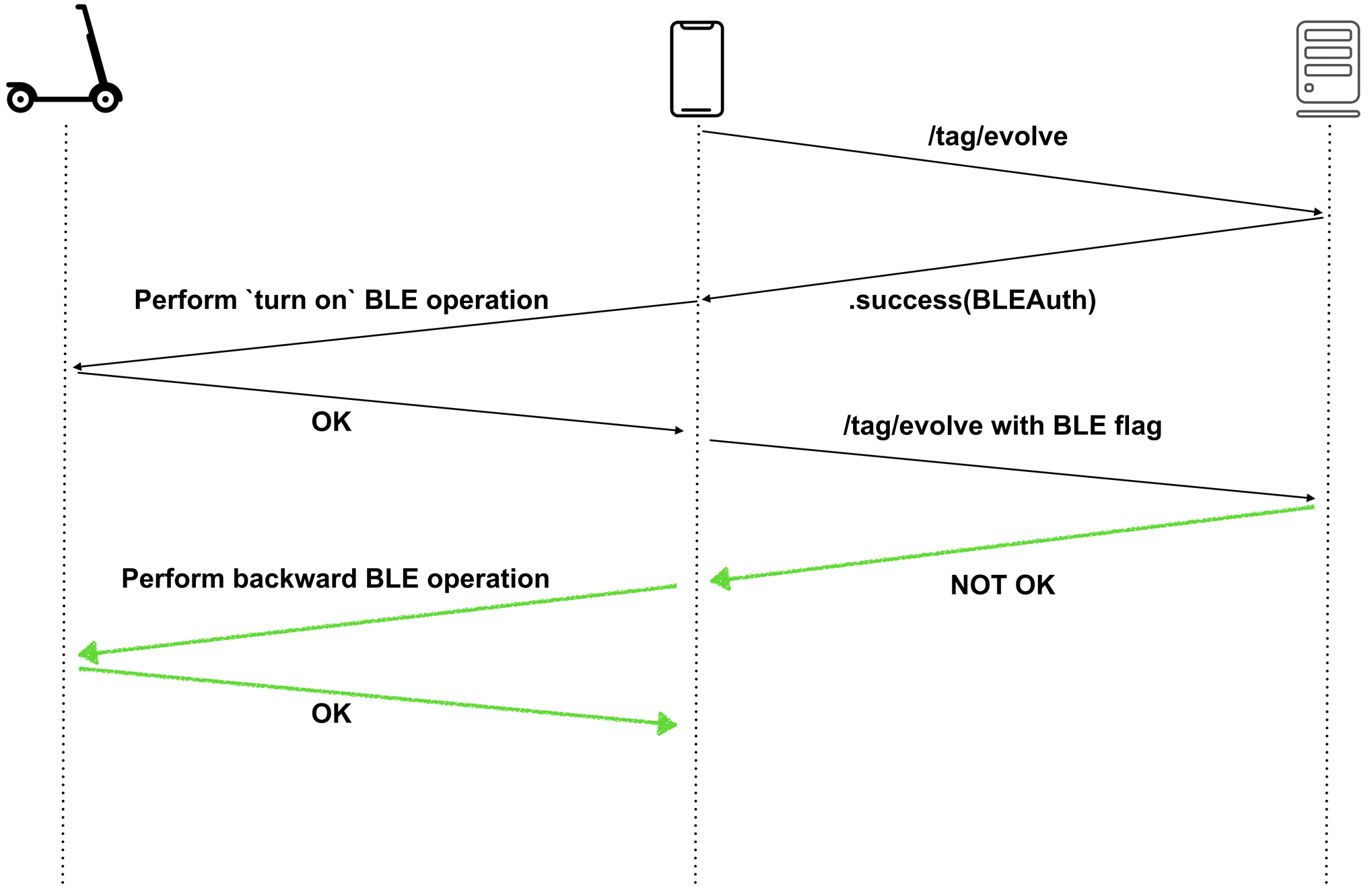
2 Избежать фрода

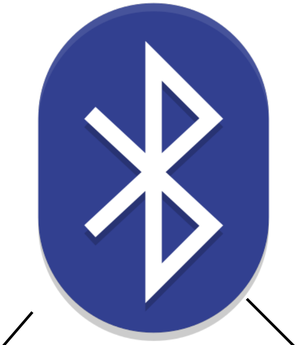

/tag/evolve
.success(BLEAuth)
Perform `turn on` BLE operation
OK
/tag/evolve with BLE flag
OK


/tag/evolve
.success(BLEAuth)
Perform `turn on` BLE operation
OK
/tag/evolve with BLE flag
OK


/tag/evolve
.success(BLEAuth)
Perform `turn on` BLE operation
OK
/tag/evolve with BLE flag
OK


/tag/evolve
.success(BLEAuth)
Perform `turn on` BLE operation
OK
/tag/evolve with BLE flag
OK


/tag/evolve
.success(BLEAuth)
Perform `turn on` BLE operation
OK
/tag/evolve with BLE flag
NOT OK
Perform backward BLE operation
OK



Low energy (LE)

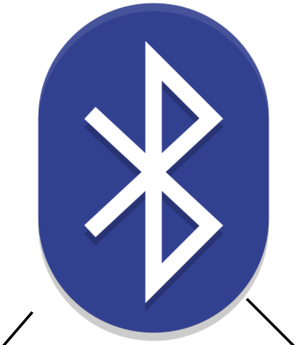
iOS 5

0.3 - 0.6 Mbps

Basic Rate / Enchanced Data Rate (BR/EDR) “classic”

iOS 13

3 Mbps



Low energy (LE)

iOS 5

0.3 - 0.6 Mbps

Basic Rate / Enchanced Data Rate (BR/EDR) “classic”

iOS 13

3 Mbps

Core Bluetooth

BLE

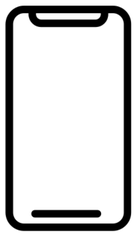
Client

Wants data

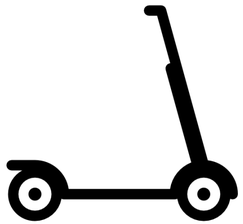
Server

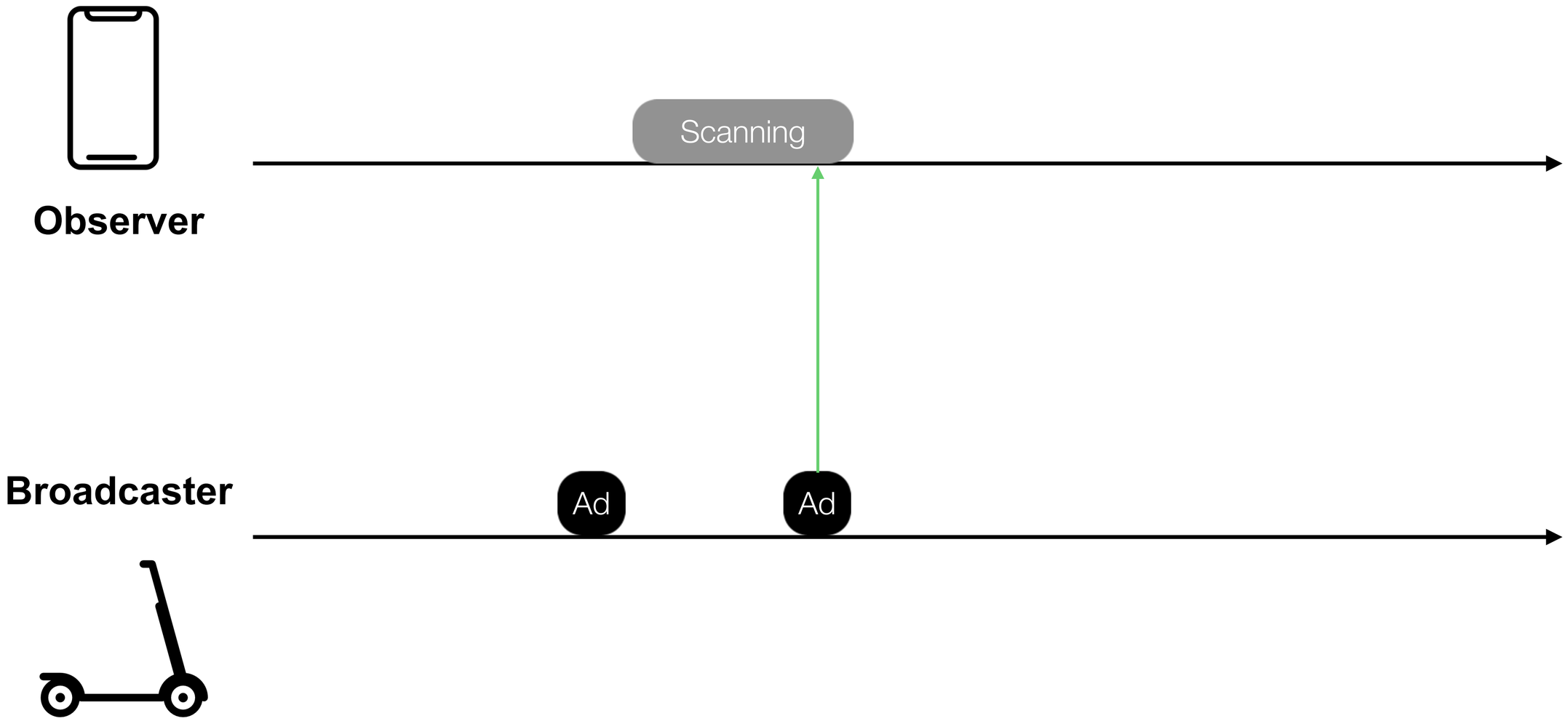
Has data

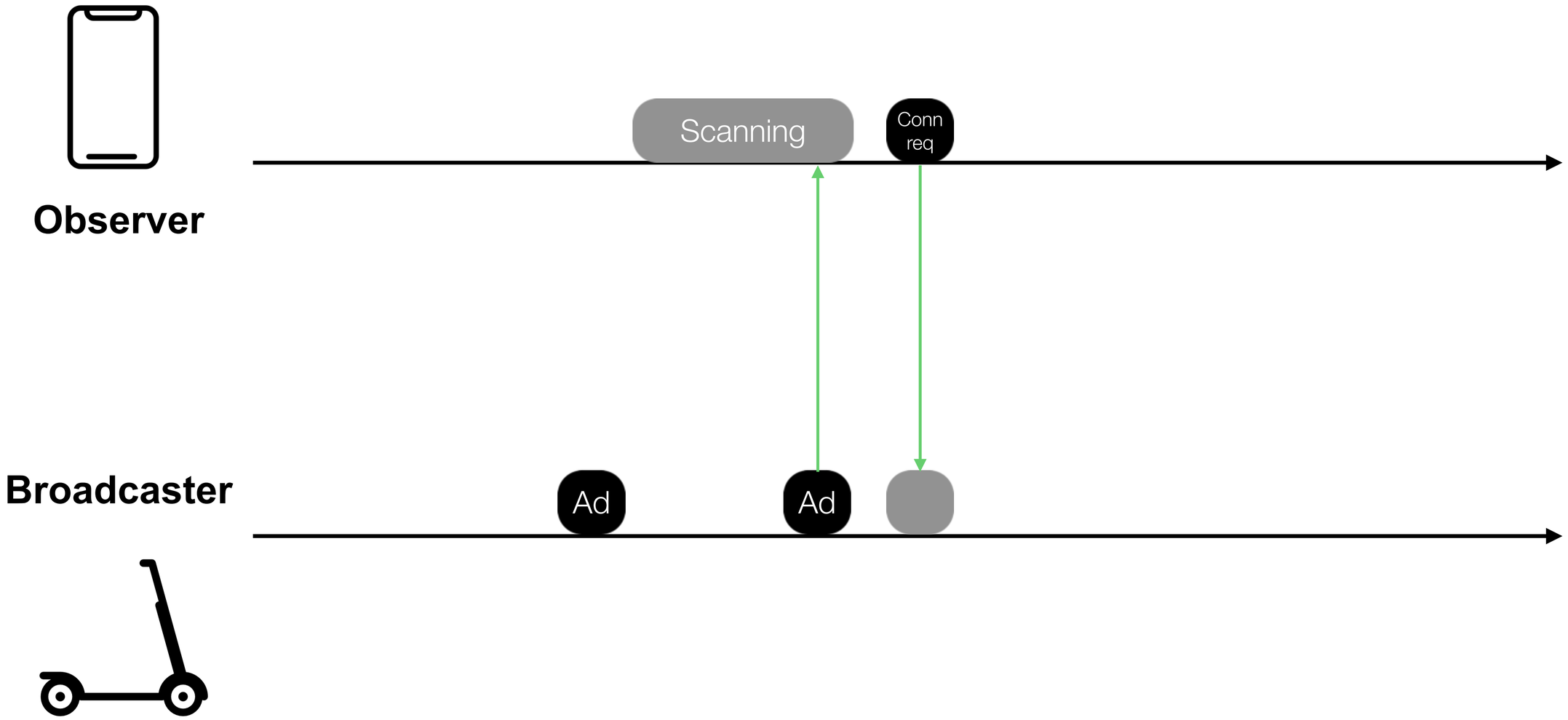
Central

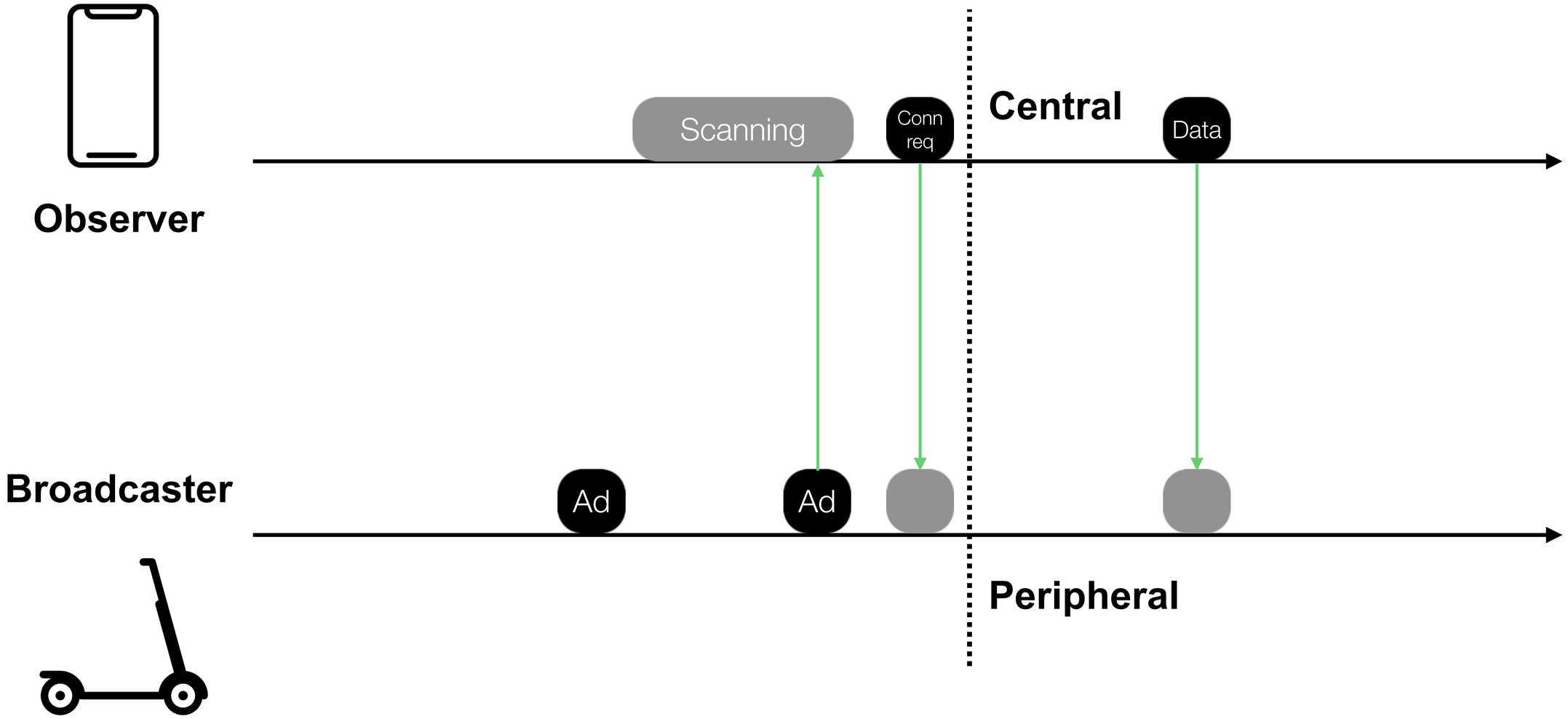


Peripheral

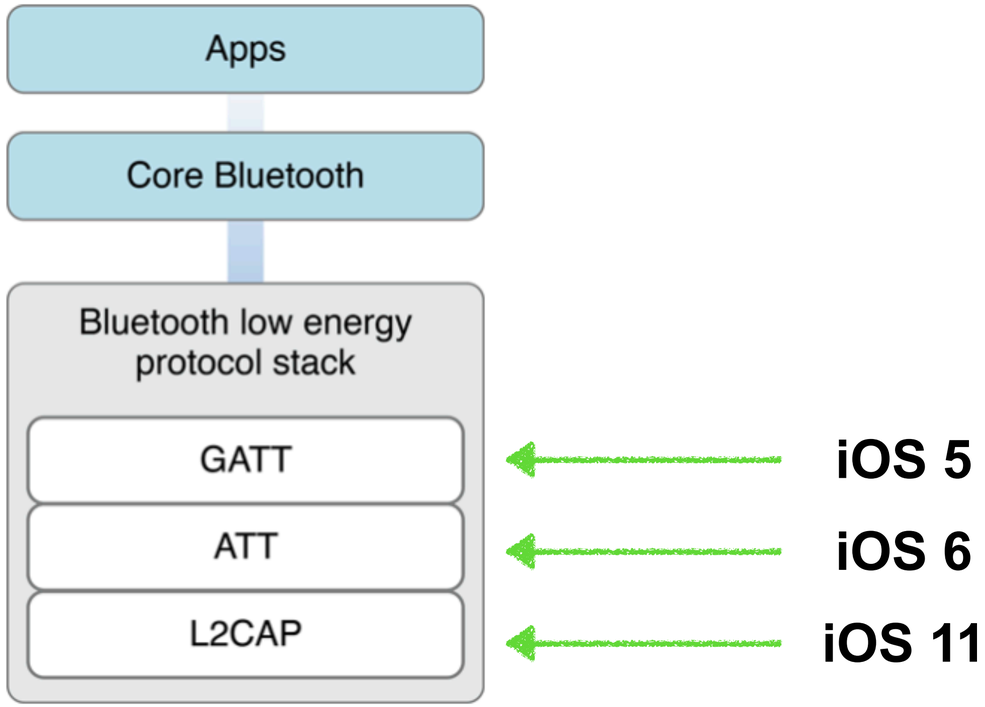



Scanning
Observer
Broadcaster
Ad
Ad

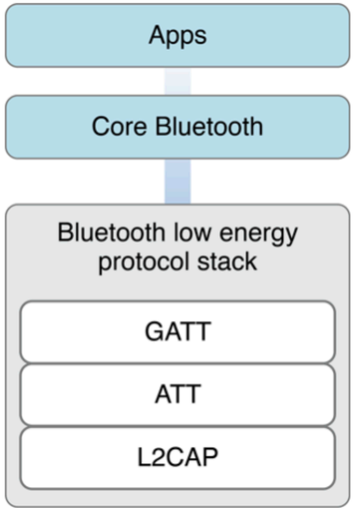

Observer
Scanning
Conn
req
Broadcaster
Ad
Ad


Observer
Scanning
Conn
req
Central
Data
Broadcaster
Ad
Ad
Peripheral

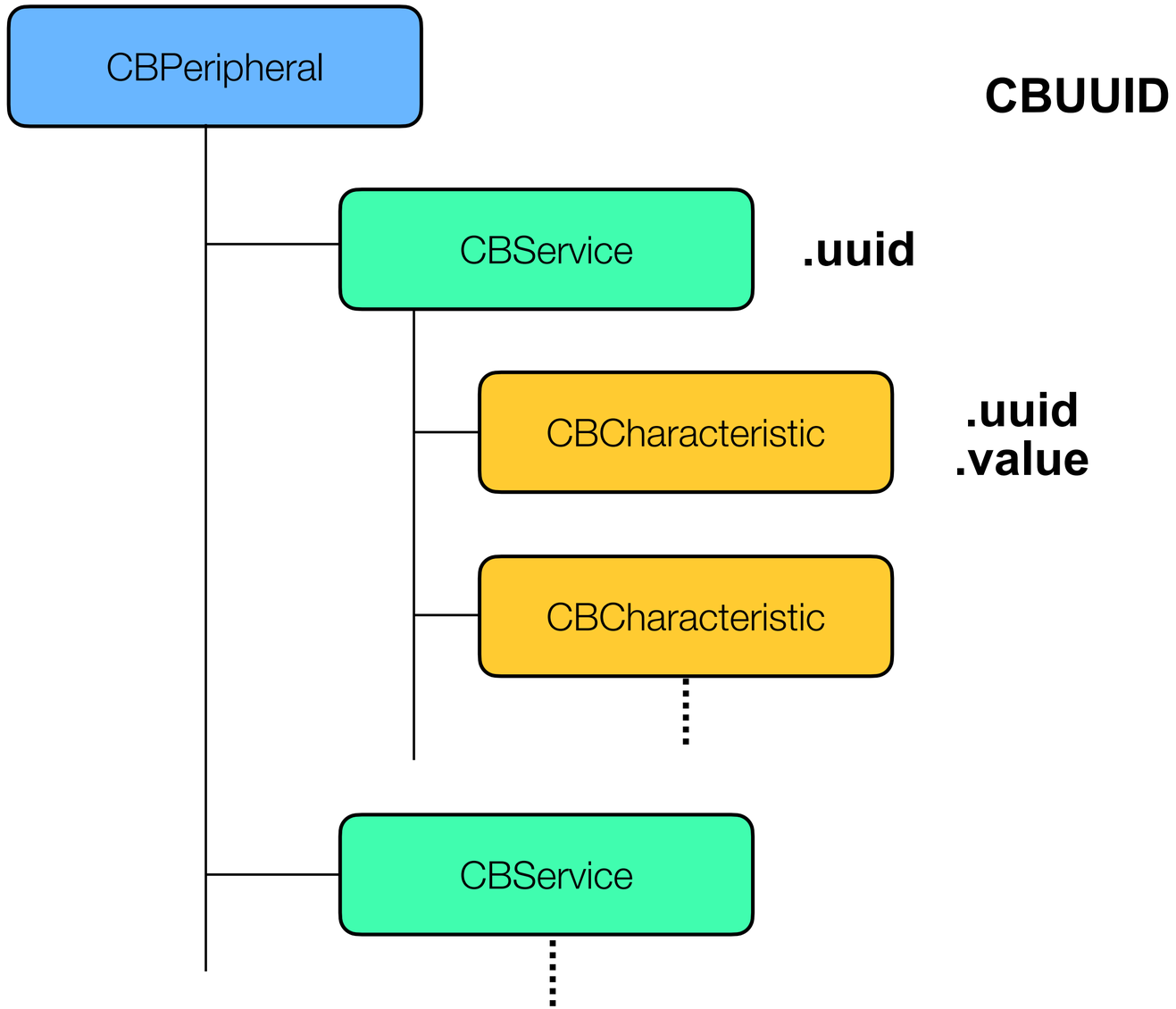
Core Bluetooth



GATT


Apps
Core Bluetooth
Bluetooth low energy protocol stack
GATT
ATT
L2CAP

iOS 5


CBPeripheral
CBUUID
CBService
.uuid
CBCharacteristic
.uuid
.value
CBCharacteristic
CBService

Сканирование (central - side)

```
let centralManager = CBCentralManager(
    delegate: self,
    queue: nil
)
```

Сканирование (central - side)

```
let centralManager = CBCentralManager(
    delegate: self,
    queue: nil
)
```

```
if manager?.state == .poweredOn {
    manager?.scanForPeripherals(
        withServices: [Constants.serviceUUID],
        options: nil
    )
}
```

Сканирование (central - side)

```
let centralManager = CBCentralManager(
    delegate: self,
    queue: nil
)
```

```
if manager?.state == .poweredOn {
    manager?.scanForPeripherals(
        withServices: [Constants.serviceUUID],
        options: nil
    )
}
```

```
func centralManager(
    _ central: CBCentralManager,
    didDiscover peripheral: CBPeripheral,
    advertisementData: [String: Any],
    rssi RSSI: NSNumber
) {
    peripheral.delegate = self
    foundedPeripherals.append(peripheral)
    manager?.connect(peripheral, options: nil)
}
```

Сканирование (central - side)

```
let centralManager = CBCentralManager(
    delegate: self,
    queue: nil
)
```

```
if manager?.state == .poweredOn {
    manager?.scanForPeripherals(
        withServices: [Constants.serviceUUID],
        options: nil
    )
}
```

```
func centralManager(
    _ central: CBCentralManager,
    didDiscover peripheral: CBPeripheral,
    advertisementData: [String: Any],
    rssi RSSI: NSNumber
) {
    peripheral.delegate = self
    foundedPeripherals.append(peripheral)
    manager?.connect(peripheral, options: nil)
}
```

```
func centralManager(
    _ central: CBCentralManager,
    didConnect peripheral: CBPeripheral
) {
    peripheral.discoverServices([Constants.serviceUUID])
}
```

Получение сервисов и характеристик (central-side)

```
func peripheral(
    _ peripheral: CBPeripheral,
    didDiscoverServices error: Error?
) {
    peripheral.services?.forEach { service in
        peripheral.discoverCharacteristics(
            [
                Constants.readCharacteristicUUID,
                Constants.writeCharacteristicUUID
            ],
            for: service
        )
    }
}
```

Получение сервисов и характеристик (central-side)

```swift
func peripheral(
    _ peripheral: CBPeripheral,
    didDiscoverServices error: Error?
) {
    peripheral.services?.forEach { service in
        peripheral.discoverCharacteristics(
            [
                Constants.readCharacteristicUUID,
                Constants.writeCharacteristicUUID
            ],
            for: service
        )
    }
}
```

```swift
func peripheral(
    _ peripheral: CBPeripheral,
    didDiscoverCharacteristicsFor service: CBService,
    error: Error?
) {
    service.characteristics?.forEach { characteristic in
        switch characteristic.uuid {
        case Constants.readCharacteristicUUID:
            readCharacteristic = characteristic
        case Constants.writeCharacteristicUUID:
            writeCharacteristic = characteristic
        default:
            break
        }
    }

    guard let r = readCharacteristic, let  = writeCharacteristic else {
        readCharacteristic = nil
        writeCharacteristic = nil
        return
    }

    /*
    peripheral.readValue(for: r)
    peripheral.writeValue(Data(), for: w, type: .withResponse)
    peripheral.writeValue(Data(), for: w, type: .withoutResponse)
     */

    peripheral.setNotifyValue(true, for: r)
}
```

Уведомление об изменении характеристики (central-side)

```
peripheral.setNotifyValue(true, for: r)
```

Уведомление об изменении характеристики (central-side)

```
peripheral.setNotifyValue(true, for: r)
```

```
func peripheral(
    _ peripheral: CBPeripheral,
    didUpdateNotificationStateFor characteristic: CBCharacteristic,
    error: Error?
) {
    guard error == nil, characteristic.isNotifying else {
        return
    }
    // ...
}
```

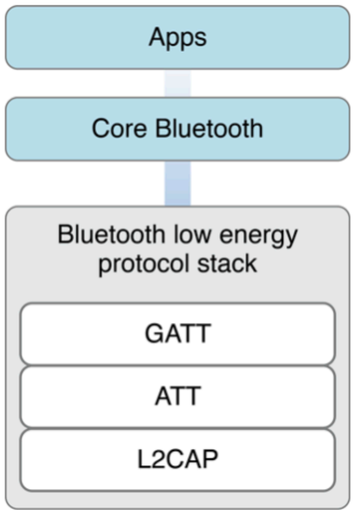
Уведомление об изменении характеристики (central-side)

```
peripheral.setNotifyValue(true, for: r)
```

```
func peripheral(
    _ peripheral: CBPeripheral,
    didUpdateNotificationStateFor characteristic: CBCharacteristic,
    error: Error?
) {
    guard error == nil, characteristic.isNotifying else {
        return
    }
    // ...
}
```

```
func peripheral(
    _ peripheral: CBPeripheral,
    didUpdateValueFor characteristic: CBCharacteristic,
    error: Error?
) {
    guard
        characteristic.uuid == Constants.readCharacteristicUUID,
        let value = characteristic.value,
        let response = Response(value)
    else { return }

    handleResponse(response, from: peripheral)
}
```

ATT



iOS 6

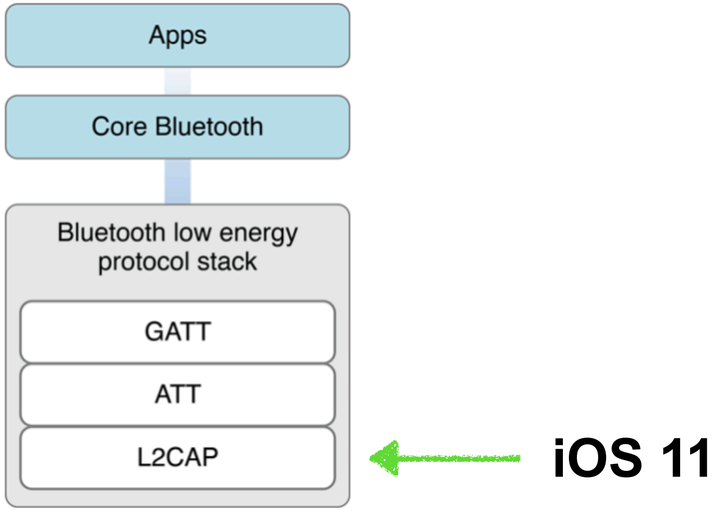
peripheral-side

```
func peripheralManager(
    _ peripheral: CBPeripheralManager,
    didReceiveRead request: CBATTRequest
) {
    if
        request.characteristic.uuid == readCharacteristic.uuid,
        let value = readCharacteristic.value
    {
        guard request.offset <= value.count else {
            peripheralManager.respond(
                to: request,
                withResult: .invalidOffset
            )
            return
        }

        request.value = value.subdata(in: request.offset..<value.count)
        peripheralManager.respond(
            to: request,
            withResult: .success
        )
    } else {
        peripheralManager.respond(
            to: request,
            withResult: .invalidAttributeValueLength
        )
    }
}
```

L2CAP

peripheral-side

```
peripheralManager.publishL2CAPChannel(withEncryption: true)
```

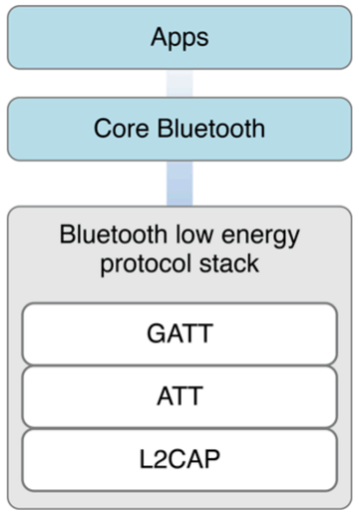


iOS 11

L2CAP

peripheral-side

```
peripheralManager.publishL2CAPChannel(withEncryption: true)
```



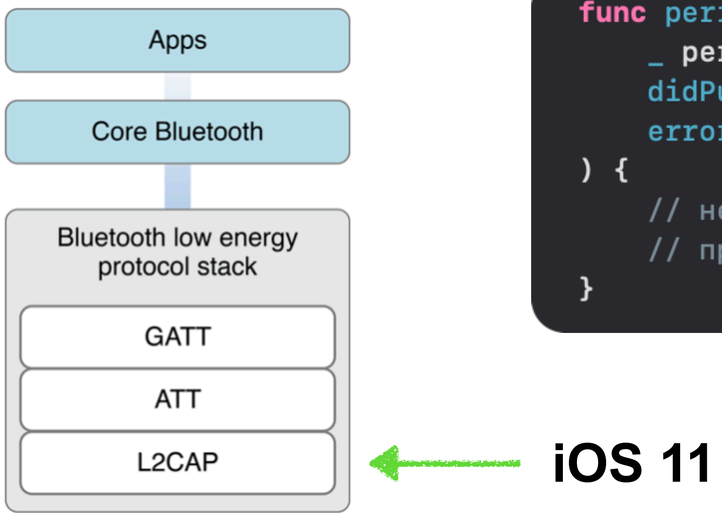

```
func peripheralManager(
    _ peripheral: CBPeripheralManager,
    didPublishL2CAPChannel PSM: CBL2CAPPSM,
    error: Error?
) {
    // необходимо отправить PSM на central через характеристику
    // при получении на стороне central делается вызов peripheral.openL2CAPChannel(_ PSM: CBL2CAPPSM)
}
```

iOS 11

L2CAP

peripheral-side

```
peripheralManager.publishL2CAPChannel(withEncryption: true)
```




```
func peripheralManager(
    _ peripheral: CBPeripheralManager,
    didPublishL2CAPChannel PSM: CBL2CAPPSM,
    error: Error?
) {
    // необходимо отправить PSM на central через характеристику
    // при получении на стороне central делается вызов peripheral.openL2CAPChannel(_ PSM: CBL2CAPPSM)
}
```

iOS 11

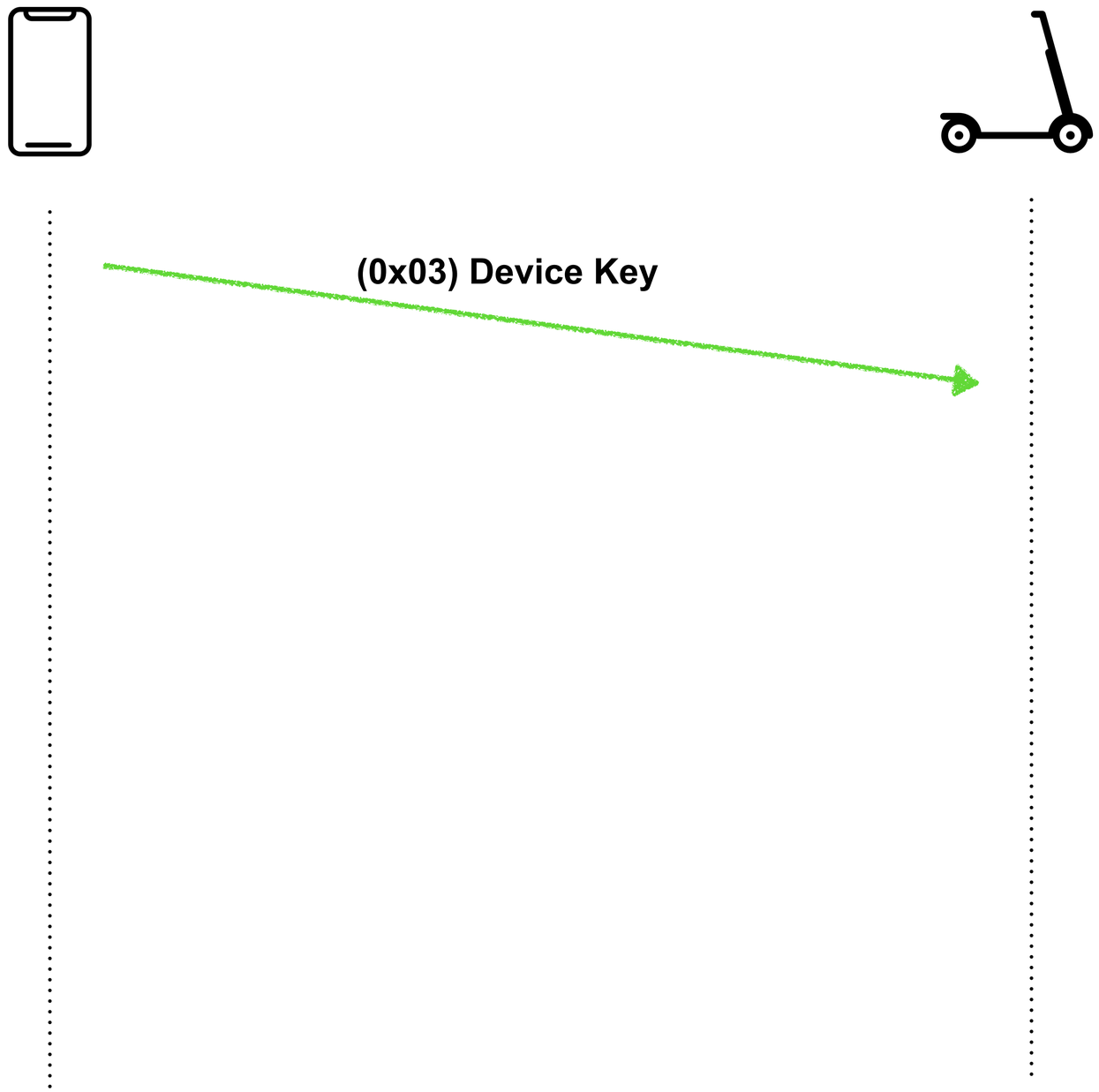
```
func peripheralManager(
    _ peripheral: CBPeripheralManager,
    didOpen channel: CBL2CAPChannel?,
    error: Error?
) {
    inputStream = channel?.inputStream
    outputStream = channel?.outputStream
}
```

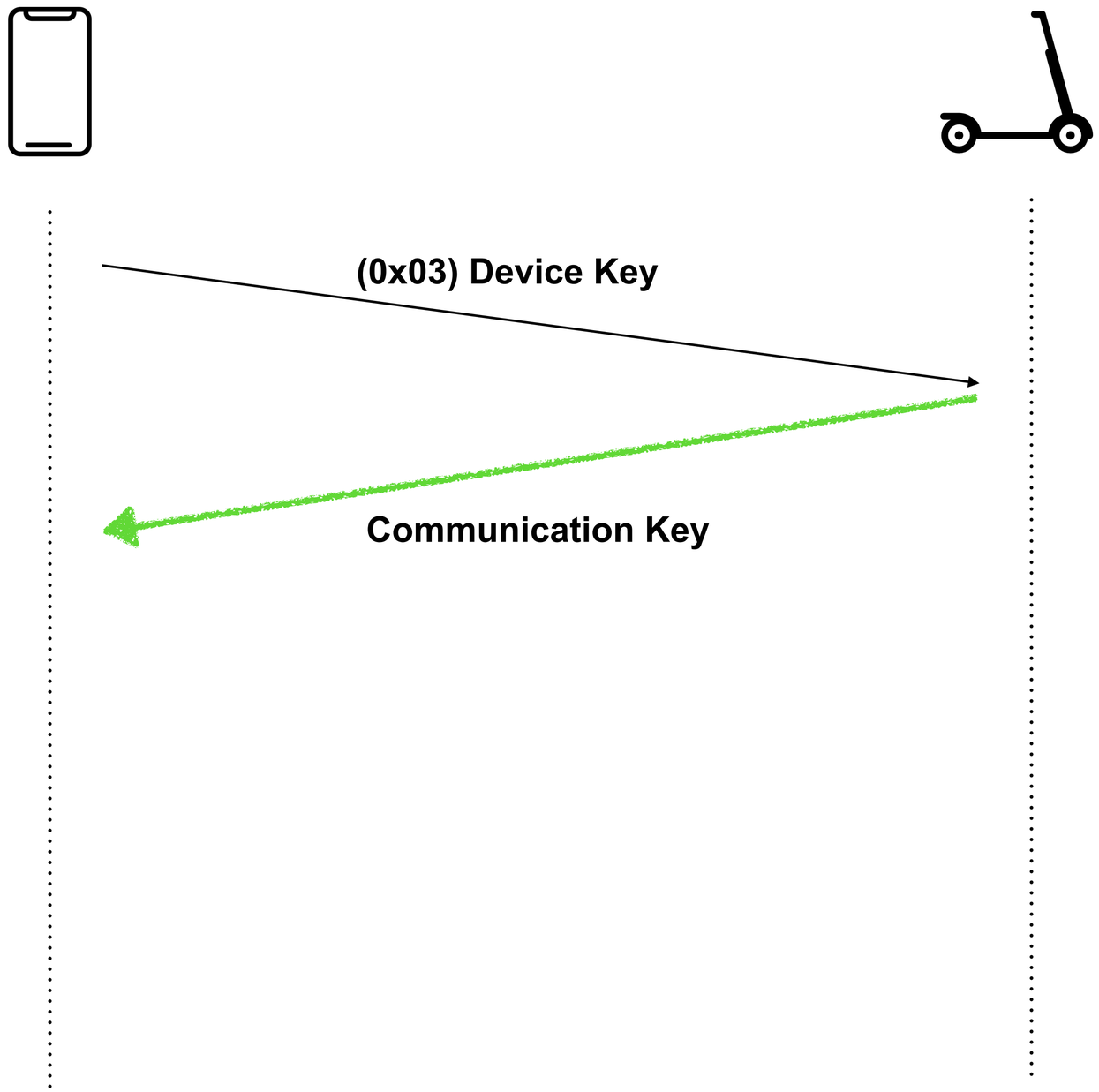
```swift
struct ScooterBLEAuth {
    typealias IMEI = String

    let name: String
    let imei: IMEI?
    let password: String
}
```

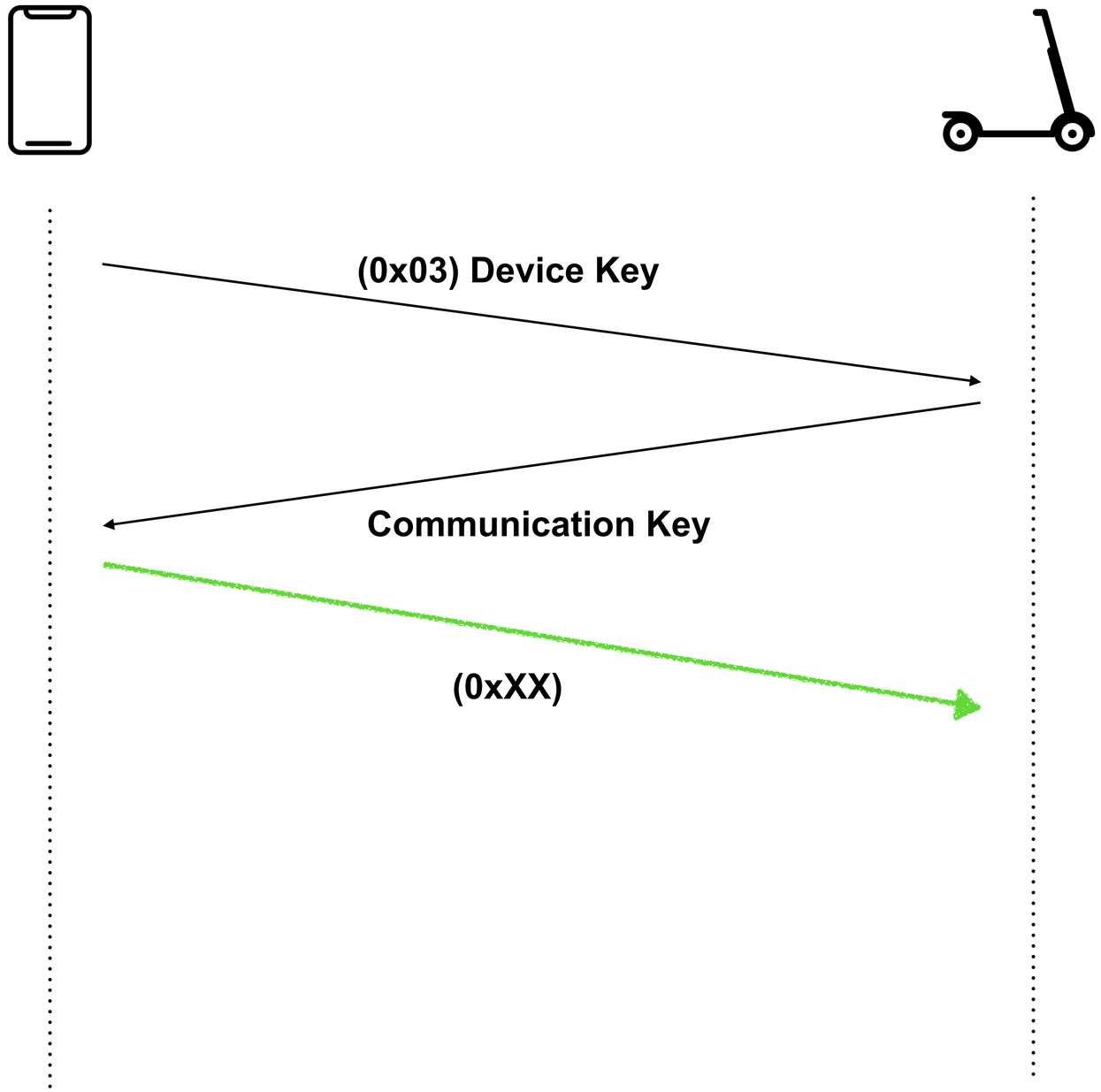
```swift
let imei = advertisementData["kCBAdvDataManufacturerData"] as? NSData
```

Протокол общения App - IoT

Bytes	Parameter
0-1	HDR
2	CMN
3	KEY
4	RND
5	LEN
6	DATA
6 + LEN	CRC

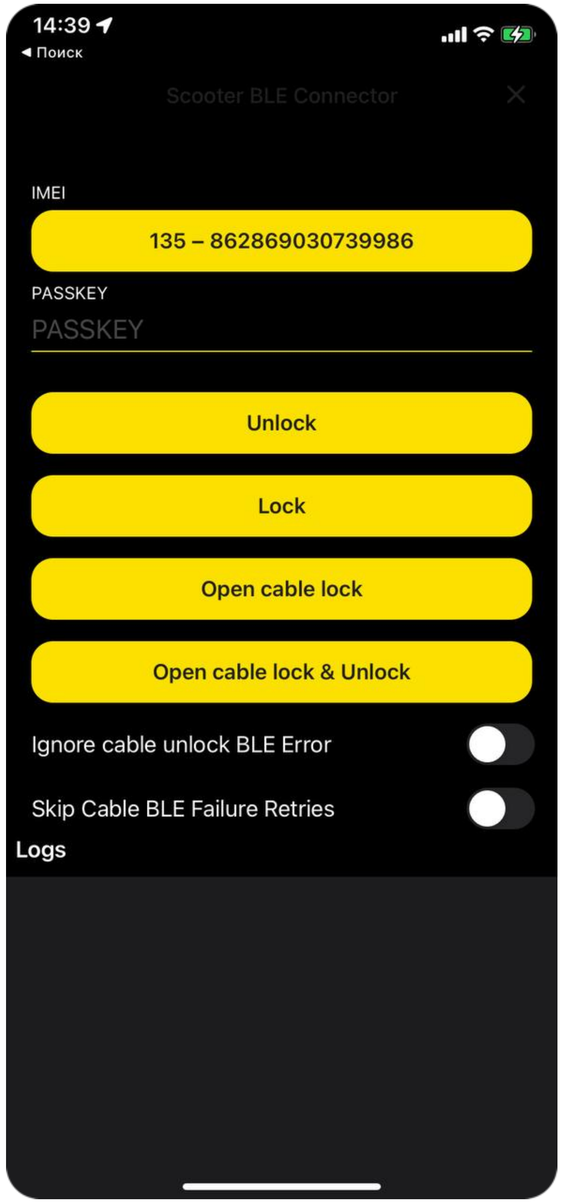

(0x03) Device Key


(0x03) Device Key
Communication Key


(0x03) Device Key
Communication Key
(0xXX)

Содержание

Тяжелый дебаг



Баги от производителя

```
private enum AuthenticationResultTag: UInt8 {
    case success = 0x1
    case failure = 0x0
}
```

```
{
    "ble_config": {
        "ignore_cable_unlock_error": true,
        "skip_cable_unlock_failure_retries": true,
    }
}
```

Баги от производителя

```
private enum AuthenticationResultTag: UInt8 {
    case success = 0x1
    case failure = 0x0
}
```

```
{
    "ble_config": {
        "ignore_cable_unlock_error": true,
        "skip_cable_unlock_failure_retries": true,
    }
}
```

Новые самокаты

```
struct ScooterBLEAuth {
    typealias IMEI = String
    typealias MAC = String

    let name: String
    let imei: IMEI?
    let mac: MAC?
    let password: String
}
```

```
let imeiOrMacData = advertisementData["kCBAdvDataManufacturerData"] as? NSData
```

Содержание

1 Уменьшение ошибочных сессий на 2-3%

2 Делаем свой IoT

3 Новые фичи

Посмотреть/почитать

WWDC 2012 Session 703 - Core Bluetooth 101

WWDC 2012 Session 705 - Advanced Core Bluetooth

WWDC 2019 Session 901 - What’s New in Core Bluetooth

Core Bluetooth Programming Guide