Find BLE devices

To find BLE devices, you use the `startScan()` method. This method takes a `ScanCallback` as a parameter. You must implement this callback, because that is how scan results are returned. Because scanning is battery-intensive, you should observe the following guidelines:

- As soon as you find the desired device, stop scanning.
- Never scan on a loop, and always set a time limit on your scan. A device that was previously available may have moved out of range, and continuing to scan drains the battery.

In the following example, the BLE app provides an activity (`DeviceScanActivity`) to scan for available Bluetooth LE devices and display them in a list to the user. The following snippet shows how to start and stop a scan:

Kotlin Java

```
private BluetoothLeScanner bluetoothLeScanner = bluetoothAdapter.getBluetoothLeScanner();
private boolean scanning;
private Handler handler = new Handler();

// Stops scanning after 10 seconds.
private static final long SCAN_PERIOD = 10000;

private void scanLeDevice() {
    if (!scanning) {
        // Stops scanning after a predefined scan period.
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                scanning = false;
                bluetoothLeScanner.stopScan(leScanCallback);
            }
        }, SCAN_PERIOD);

        scanning = true;
        bluetoothLeScanner.startScan(leScanCallback);
    } else {
        scanning = false;
        bluetoothLeScanner.stopScan(leScanCallback);
    }
}
```

Note: The `BluetoothLeScanner` is only available from the `BluetoothAdapter` if Bluetooth is currently enabled on the device. If Bluetooth is not enabled, then `getBluetoothLeScanner()` returns null.

To scan for only specific types of peripherals, you can instead call `startScan(List<ScanFilter>, ScanSettings, ScanCallback)` , providing a list of `ScanFilter` objects that restrict the devices that the scan looks for and a `ScanSettings` object that specifies parameters about the scan.

The following code sample is an implementation of `ScanCallback` , which is the interface used to deliver BLE scan results. When results are found, they are added to a list adapter in the `DeviceScanActivity` to display to the user.

Kotlin Java

```
private LeDeviceListAdapter leDeviceListAdapter = new LeDeviceListAdapter();

// Device scan callback.
private ScanCallback leScanCallback =
        new ScanCallback() {
            @Override
            public void onScanResult(int callbackType, ScanResult result) {
                super.onScanResult(callbackType, result);
                leDeviceListAdapter.addDevice(result.getDevice());
                leDeviceListAdapter.notifyDataSetChanged();
            }
        };
```

Note: You can only scan for Bluetooth LE devices *or* scan for classic Bluetooth devices, as described in Bluetooth overview. You cannot scan for both Bluetooth LE and classic devices at the same time.