**Attention:** **This description document is written in our case, if your application requires other functions, you can parse the data yourself, the specific parsing process has been given in the Eddystone class and https://github.com/google/eddystone/blob/master/protocol-specification.md**

**1.** By integrating the Application class to implement our own Application, In the onCreate () method, the URL prefix and suffix in the Eddystone-URL mode is initialized. as follows：

**public** **void** onCreate() {

**super**.onCreate();

*hashMap* = **new** HashMap<String, String>();

*hashMap*.put("00", ".com/");

*hashMap*.put("01", ".org/");

*hashMap*.put("02", ".edu/");

*hashMap*.put("03", ".net/");

*hashMap*.put("04", ".info/");

*hashMap*.put("05", ".biz/");

*hashMap*.put("06", ".gov/");

*hashMap*.put("07", ".com");

*hashMap*.put("08", ".org");

*hashMap*.put("09", ".edu");

*hashMap*.put("0a", ".net");

*hashMap*.put("0b", ".info");

*hashMap*.put("0c", ".biz");

*hashMap*.put("0d", ".gov");

*hashMapPro* = **new** HashMap<String, String>();

*hashMapPro*.put("00", "http://www.");

*hashMapPro*.put("01", "https://www.");

*hashMapPro*.put("02", "http://");

*hashMapPro*.put("03", "https://");

}

**2.** In the callback of the scan to the Bluetooth, we put the Bluetooth broadcast data into Eddystone object. as follows：

**final** Eddystone *eddystone* = EddystoneClass.fromScanData(device, rssi, scanRecord);

**3.** In the Eddystone class which has a specific implementation of the broadcast data analysis of Bluetooth devices. Specific analysis can be found in <https://github.com/google/eddystone/blob/master/protocol-specification.md>，You can also parse the data yourself.

**4.** There is a hybrid adapter in the LeDeviceListAdapter class, which displays different data in different types, you can just write the adapter as long as you can.

**5.** If there is something you don't understand, you can refer to our example.