The Bluetooth® Beacon Primer

Release: 1.3.3

Document Version : 1.3.3

Last Updated: October 17th 2022

Contents

Revision History

Version	*Date*	*Author*	*Changes*
1.0.0	1st August 2018	Martin Woolley, Bluetooth SIG	Initial version
1.3.1	27th October 2020	Martin Woolley, Bluetooth SIG	Format update and brand compliance changes.
1.3.2	21st December 2020	Martin Woolley Bluetooth SIG	Language changes
1.3.3	17th October 2022	Martin Woolley Bluetooth SIG	Added reference to the Bluetooth Low Energy Primer resource.

1. Introduction

Bluetooth® beacons are big news, with 2014 the year when many organizations ran their initial pilot projects and 2015 the year when companies rolled out large deployments of beacons to create major new services. In fact ABI Research predict we'll see a 60 million unit market by 2019. That's a whole lot of beacons.

In the majority of cases, beacons are being used to provide a way of determining a person's location when inside a building such as a department store, airport, office or museum and to provide a service which leverages that location information. In these scenarios the beacons are installed in fixed locations and the person's smart phone has an application on it which knows the location of each of the beacons it is intended to work with. Interestingly though we're beginning to see a new class of beacon application where the beacons are mobile rather than stationary. Take a look at the Lighthouse solution (http://www.lighthouseforkids.co/), a wearable beacon device designed to make it possible to keep track of students with special needs and keep them safe. We think we'll see more solutions involving mobile and wearable beacons like this in the future.

Bluetooth beacons are one of many applications of Bluetooth Low Energy (LE).

For an introduction to technicalities and terminology of Bluetooth Low Energy, download the Bluetooth Low Energy Primer from https://www.bluetooth.com/bluetooth-resources/the-bluetooth-low-energy-primer/ .

Topics such as the Generic Access Profile (GAP) and advertising are of particular relevance to beacons.

2. Commercial Beacon Products

Figure 1 - Examples of Commercial Beacon Products

There are a multitude of commercial beacons on the market and as the graphic shows, they come in all sorts of shapes and sizes. They all serve the same fundamental purpose and exploit the same aspects of Bluetooth LE. That said there have emerged a few slightly different approaches to the way beacons work and this has implications for the range of use cases a given beacon product may or may not be able to address and for developers who create applications which work with beacons.

3. Advertising Works

Figure 2 - Bluetooth LE Device Discovery and Advertising

The Bluetooth LE stack includes a component called the Generic Access Profile which we call “GAP” for short. GAP is responsible for the procedure that allows devices to discover each other and then connect. The discovery process involves devices either “advertising” or “scanning”. Advertising devices broadcast small packets of data at regular intervals and those packets contain information which help scanning devices to decide whether or not the advertising device is of a relevant type which it might want to connect to.

Beacons advertise but generally speaking, other devices do not connect to them. Beacons are said to be *GAP Broadcasters* which is the technical term for a Bluetooth LE device which advertises but which does not accept connections. The reason beacons are GAP Broadcasters is that everything required to accomplish the goal of allowing a smartphone application to determine its location is contained within the advertising packet. So what exactly is in an advertising packet emitted by a beacon?

4. Beacon Frame Formats

Advertising packets can contain a small number of data fields of various types. There’s a limit of 31 octets (bytes) available for data and so product designers have to choose the fields they want to include

in advertising packets carefully[1]. There are a number of field types called AD Types and these are defined by the Bluetooth SIG in a document called the Core Specification Supplement (CSS). All AD Types have the same structure, with three parts. The first is a length byte, the second is a byte indicating the type and the third part is the value.

Beacons encode information in one or more of the standard advertising packet fields but there are several schemes in use today and we call these variations *frame formats*. The three primary formats are iBeacon from Apple, AltBeacon from Radius Networks and Eddystone from Google.

Application developers creating beacon applications need to know the frame format used by the beacons their application must work with and be able to decode it.

4.1 Frame Format: iBeacon

Perhaps the best known of the three frame formats described in this paper, is Apple's iBeacon. iBeacon uses a standard AD Type called the Manufacturer Data field which has type value 0xFF. An iBeacon advertising packet is shown below in raw format and then with each individual AD Type listed in the table.

Raw data:

0x0201061AFF4C000215E2C56DB5DFFB48D2B060D0F5A71096E000000000C3

Details:

LEN.	TYPE	VALUE
2	0x01	0x06
26	0xFF	0x4C000215E2C56DB5DFFB48D2B060D0F5A71096E000000000C3

Figure 3 - iBeacon advertising packet

Everything an application needs to identify a specific iBeacon and derive useful information about it is in the Manufacturer Data field. Here's how its value breaks down:

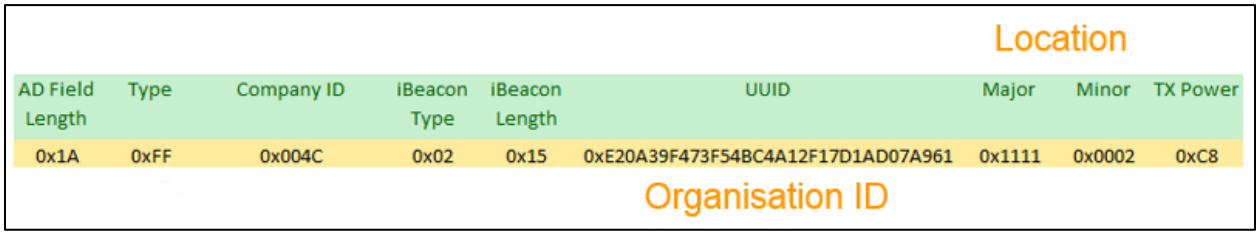


Figure 4 - Break down of the Manufacturer Data field carrying iBeacon data

[1] Extended Advertising, introduced in Bluetooth core specification version 5 allows much larger payloads to be advertised

The Manufacturer Data field must always start with a 2 byte company identifier code. These codes are issued to companies by the Bluetooth SIG and indicate to applications how to decode the remainder of the field. The value 0x004C shown in Figure 4 is Apple's Company ID.

iBeacon Type indicates the device to be a *proximity beacon* and this always has a value of 0x02 for iBeacon devices. The iBeacon length field which follows says how many bytes the remainder of the field contains.

The UUID identifies the organisation that owns the beacon.

The Major and Minor field encode a location with the Major field usually indicating a specific building and the Minor field a particular location within that building. So between them these fields might be interpreted as meaning *Central London store, Sporting Goods department*.

The TX Power field is there to help applications estimate the distance the user is from the beacon. I'll say more about this below but in the case of iBeacon, it contains the signal strength as measured at 1 metre from the beacon.

See https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf

4.2 Frame Format: AltBeacon

The AltBeacon frame format is almost identical to the iBeacon frame format. The two formats are presented side by side in Figure 5.

AD Field Length	Type	Company ID	Beacon Code	Beacon ID org. unit	Beacon ID (remainder)	TX Power
				AltBeacon		
0x1A	0xFF	0x0118	0xBEAC	0xE20A39F473F54BC4A12F17D1AD07A961	0x11110002	0xC8

AD Field Length	Type	Company ID	iBeacon Type	iBeacon Length	UUID	Major	Minor	TX Power
					iBeacon			
0x1A	0xFF	0x004C	0x02	0x15	0xE20A39F473F54BC4A12F17D1AD07A961	0x1111	0x0002	0xC8

Figure 5 - AltBeacon compared with iBeacon

An AltBeacon frame will usually have the company ID of Radius Networks, the company that invented the AltBeacon frame format. It also defines a Beacon Code which contains the cheekily selected hex value 0xBEAC instead of the iBeacon type and length field. A UUID is used to identify the organisation which owns the beacon as per iBeacon. Rather than break the location code down into two parts, AltBeacon lets you use those 4 bytes any way you like. And the TX Power field at the end is the same as with iBeacon.

See http://altbeacon.org/

4.3 Frame Format: Eddystone

Google's Eddystone works a little differently to iBeacon and AltBeacon. It doesn't use the Manufacturer Data field in advertising packets at all. Instead it uses the *Complete List of 16-bit Service UUIDs* field and the associated Service Data field. Specifically, a service UUID of 0xFEAA appears in the service UUIDs field and the Service Data field contains 0xFEAA followed by the service data i.e. the beacon data.

Furthermore, Eddystone defines an extensible list of sub-types as shown in

Type ID	Name	Comments
0x00	UID	Unique 16-byte Beacon ID composed of a 10-byte namespace ID and a 6-byte instance ID
0x10	URL	URL using a compressed encoding format
0x20	TLM	Telemetry information about the beacon itself such as battery voltage, device temperature, and counts of broadcast packets

See https://github.com/google/eddystone

Next I'll discuss some of the issues which beacon application developers may need to think about and address.

5. Beacon Applications

5.1 Dedicated Application or Generalised Beacon Browser?

With the possible exception of Eddystone, a given set of beacons, deployed by a particular organisation for a particular purpose will require a corresponding application to recognise those beacons and respond to them in some way. So for example an application designed to work with beacons deployed across all branches of a national chain of stores might need to recognise beacons from those stores, determine the location signified by the data in the advertising packet and then pop up a special offer to the user relating to the department and store they are in, assuming it is something this user might be interested in. Or an airport might deploy beacons to allow their application to report to a server whenever a user passes a particular beacon. That way the system can track users as they pass through the airport and lend assistance, for example if the user is stuck in security and at risk of missing their flight.

There will be a need for a dedicated application whenever beacons broadcast unique codes such as UUID/Major/Minor data with the intention that an application map this data to something else like a location and take a custom action such as presenting that special offer to the user.

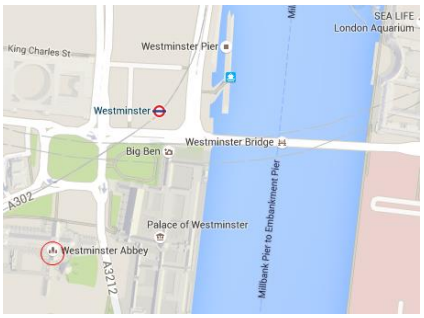
Eddystone allows URLs to be broadcast as an alternative to a unique ID. A URL can be consumed and acted upon by something like a browser and therefore it's easy to envisage a generalised beacon application which can process URLs broadcast by *any* beacon, not just the set deployed by one particular organisation.

5.2 APIs

Developers have two broad choices where APIs are concerned. They can either use the native Bluetooth LE APIs provided by their platform and implement periodic scanning and filtering designed to capture advertising packets from the beacons their application is interested in or it might be an option to use a higher level, dedicated beacon API from the beacon manufacturer or frame format specifier. Radius Networks provide an Android library which implements an AltBeacon API for example. Which approach you take is a personal choice and you'll probably weigh up the pros and cons of having complete control vs perhaps getting the job done a bit faster.

5.3 Where Am I?

The answer to this question can be expressed in various ways according to your requirement. In each case though, don't forget that the question more correctly formulated should be "Where's this beacon". Where the *user* specifically is, relative to a given beacon is more complicated. The answer might be a precise GPS location or it might be a location expressed in a descriptive way like *The sporting goods department of the London store*. Alternatively the location might be expressed in terms of proximity to something else like *By the Tyrannosaurus Rex exhibit*.



With iBeacon and AltBeacon you're not given this location data directly. You need to map the UUID/Major/Minor (iBeacon) or UUID/Beacon Code (AltBeacon) to that location data somehow. There are various ways you could approach this, the obvious ones being to

1. Perform a look up against a local database on the smartphone

2. Perform a look up against a database on a remote server

3. Reference a local database but keep the content of that database synchronised against a remote server copy whenever network conditions are suitable.

However you decide to do it, the essence of the task is the same.

5.4 Distance Estimation

The standard approach to estimating the distance a user is from a beacon is to take the signal strength of the beacon as measured and reported by the smartphone and perform a path loss calculation. The signal strength is called the RSSI, the Received Signal Strength Indication. A path loss calculation will make use of both this value and the TX Power value included within the advertising packet since this acts as a reference value and gives us the expected signal strength when a certain distance from the beacon (one metre with iBeacon, zero metres with Eddystone) .

Wikipedia has an article on path loss calculations: https://en.wikipedia.org/wiki/Free-space_path_loss

This approach will yield only a rough estimate however and the environment the beacon is in may have a very significant effect on distance estimation. A weaker signal strength could be caused by the user walking behind a pillar in the museum rather than them having moved further away from the beacon for example.

A more advanced approach is to use a technique called Radio Fingerprinting where a radio map of the location is created and this data leveraged to produce better results.

The International Computer Science Institute have published a paper on Radio Fingerprinting:

http://www.icsi.berkeley.edu/pubs/speech/preciseindoor10.pdf

5.5 Scanning

Beacon applications will scan periodically for beacons and will implement filtering so that only devices of interest are examined more closely. Scanning is a relatively power hungry operation, even with Bluetooth LE and so very aggressive scanning spanning long periods of time should be avoided if possible.

You have two basic parameters to play with when setting up scanning. The “scan window” which specifies how long to scan for and the “interval” which is how long to wait in between scans. APIs on various platforms may express this differently but this is what it all boils down to.

When thinking about scan window and interval values, you should consider how often the beacons will be advertising. There’s no point scanning for 500ms seconds with a 100ms interval if your beacons only emit an advertising packet every 2000ms. Clearly there will be many occasions when you do not receive an advertising packet and the scanning activity will have been a waste. Note though that as at version 4.2 of the Bluetooth core specification, advertising is not deterministic since small random delays are included in the scheduling algorithm to help avoid persistent collisions. Therefore you’ll need to adopt a sensible approximation to synchronising your scanning with the beacons’ advertising and ensure your code is not too quick to jump to conclusions when a beacon is no longer present in the packets received by scanning.

5.6 Disappearing Beacons

Each time your application scans, it will detect zero to many beacons. It’s likely that you will see the same beacon(s) in your scan results many times in succession since it will take considerably more time for the user to move out of range of a particular beacon than its advertising interval which is likely to be in the order of hundreds of milliseconds.

At some point though a beacon will stop appearing in your scan results and your application will need to decide how to interpret this. You could immediately conclude that the beacon is no longer in range because the user has moved away from it but this would probably be a mistake. A beacon might stop being reported for a number of reasons and its absence from one set of scan results does not mean it won’t appear again the next time you scan, a few seconds later. Perhaps the user walked behind an object which attenuated the signal from the beacon enough to make it temporarily out of range despite the user being no further away from it physically than the last time packets from the beacon were received, in which case when the user takes a few more steps forward and appears from behind the

pillar, you'll start receiving those packets again. In short the user was "near to" the beacon all the time, even when you stopped received advertising packets briefly.

Applications often maintain a list of "live" beacons, those that the user is believed to be near to. Removal from the live list could be performed immediately a beacon does not appear in the latest scan results or you could decide to defer removal until the beacon has been absent from n sets of scan results in succession or perhaps after it has been absent for a specific period of time. Either of these approaches would probably help you deliver a better experience to the users of your beacon application.

5.7 Bluetooth. Beacon. Action!

Having determined location from a beacon's data, what next? Obviously what your application now does is very application-specific. You could do something interactive like displaying information about the nearest museum exhibit to the user.

If you do go for an interactive response than it's critical you only the user relevant information and don't do so too often. Information can easily become Irritation and users do know how to uninstall applications!

Alternatively you might do something silently in the background. Tracking users through the airport so that they can be assisted if they get delayed is a good example of this. Obviously respect for the user's privacy is of paramount importance and users should explicitly opt in to something like this.

5.8 Do Not Disturb

What the user wants and likes may vary over time. When visiting the mall with the whole family, perhaps they don't want to have their phone notifying them of special offers whereas when they're shopping alone they do. It's a good idea to consider this and give the user an easy way to tell the beacon application to not disturb them until told otherwise.

5.9 Wears that Beacon

Wearables like Smart Watches are tailor made for beacon applications glancing at your wrist is much easier than pulling your phone out of your pocket to look at its screen. Smart watch applications can be completely interactive too so there's lots of scope for some very interesting wearable beacon applications.

5.10 Roll Your Own Beacon

Developers need to test their applications and this means they need beacons in their test environment. You can (and should) use beacons of the same type which you'll roll out. But ahead of this stage, it can be flexible and more productive to use a programmable device to act as your test beacon. Using a programmable device means you can quickly and easily change beacon advertising data and other parameters which govern its behaviour for testing purposes.

A device like a Raspberry Pi or Intel Edison can be turned into a beacon as can more specialised Bluetooth developer boards such as those which most of the Bluetooth chip and module manufacturers sell. Devices like the Texas Instruments SensorTag, the Nordic Thingy or a BBC micro:bit can also be turned into a beacon very easily without the need to do any programming but still giving you plenty of versatility for other Bluetooth development purposes.

6. Conclusion

Beacons are going to be a very common feature of all sorts of environments. Knowing how they work and how to exploit them is a skill any developer would benefit from having in this, the new age of the Internet of Things.