6 BLUETOOTH APPLICATION ARCHITECTURE

6.1 BLUETOOTH PROFILES

Application interoperability in the Bluetooth system is accomplished by Bluetooth profiles. Bluetooth profiles define the required functions and features of each layer in the Bluetooth system from the PHY to L2CAP and any other protocols outside of this specification. The profile defines the vertical interactions between the layers as well as the peer-to-peer interactions of specific layers between devices.

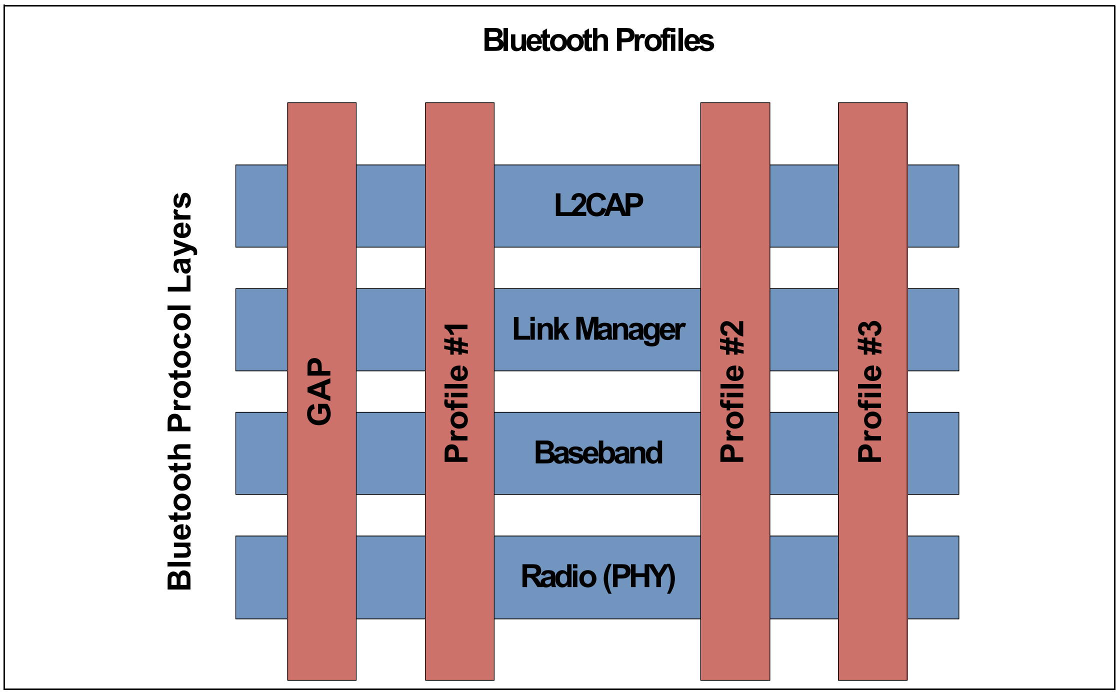

Figure 6.1: Bluetooth profiles

In addition, application behaviors and data formats are also defined by the profile. When two devices comply with all the requirements of a Bluetooth profile, interoperability of an application is enabled.

All profiles describe service discovery requirements necessary for devices to connect, find available application services and connection information necessary for making application level connections.

6.2 GENERIC ACCESS PROFILE

The Bluetooth system defines a base profile which all Bluetooth devices implement. This profile is the Generic Access Profile (GAP), which defines the basic requirements of a Bluetooth device. For instance, for BR/EDR, it defines a Bluetooth device to include the Radio, Baseband, Link Manager, L2CAP, and the Service Discovery protocol functionality; for LE, it defines the Physical

Layer, Link Layer, L2CAP, Security Manager, Attribute Protocol and Generic Attribute Profile. This ties all the various layers together to form the basic requirements for a Bluetooth device. It also describes the behaviors and methods for device discovery, connection establishment, security, authentication, association models and service discovery.

In BR/EDR, GAP defines a single role with functionality that may be present in each device. This functionality includes how devices discovery each other, establish connections and describes security association models used for authentication. In BR/EDR this functionality may be present in both devices. It may be necessary for a device to implement both the initiating and accepting functionality if the device wants to discover or establish connections with all devices. A device may only include either the initiating or the accepting functionality but it requires the remote device to support the complimentary functionality to discovery or establish connections with the device. For BR/EDR, the Controller is required to support all the functionality, however the Host may limit this functionality based on the other profiles or use cases supported by the device.

In LE, GAP defines four specific roles: Broadcaster, Observer, Peripheral, and Central. A device may support multiple LE GAP roles provided that the underlying Controller supports those roles or role combinations. Each role specifies the requirements for the underlying Controller. This allows for Controllers to be optimized for specific use cases.

The Broadcaster role is optimized for transmitter only applications. Devices supporting the Broadcaster role use advertising to broadcast data. The Broadcaster role does not support connections. The Observer role is optimized for receiver only applications. Devices supporting the Observer role are the complementary device for a Broadcaster and receives broadcast data contained in advertisements. The Observer role does not support connections. The Peripheral role is optimized for devices that support a single connection and are less complex than Centrals; it uses the Link Layer Peripheral role within the connection. The Central role supports multiple connections and is the initiator for all connections with devices in the Peripheral role; it uses the Link Layer Central role within the connection. Devices supporting the Central role generally support more complex functions compared to the other LE GAP roles.

6.3 PROFILE HIERARCHY

Since all Bluetooth devices are required to implement GAP, any additional profiles implemented by a Bluetooth device become supersets of GAP. Depending on the complexity of an application or the ability to reuse common requirements of functionality of the Bluetooth system between many applications, additional generic profiles can be created that depend on GAP or other generic profiles, as well as enabling other profiles. A top level profile that describes application interoperability is called an Application Profile.

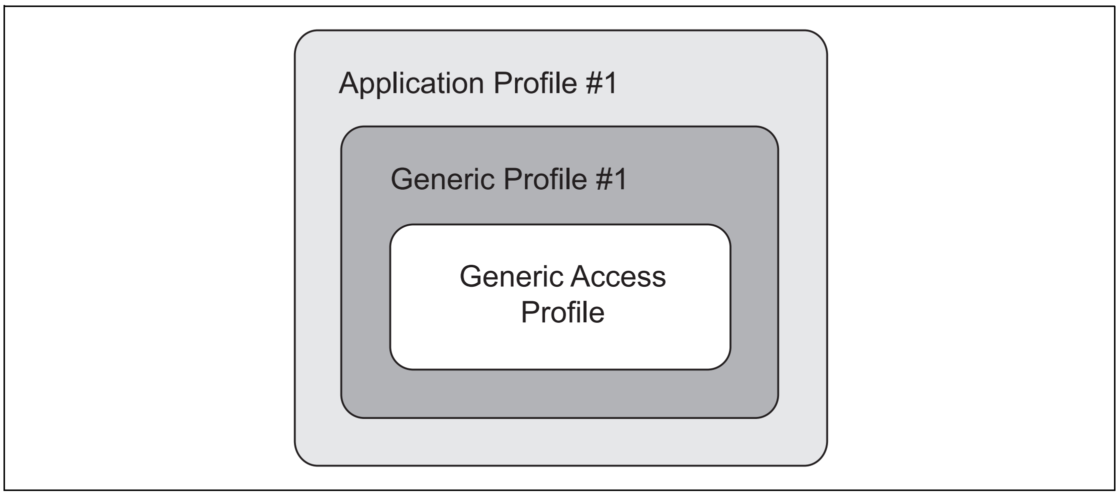

Figure 6.2: Profile hierarchy

Application profiles contain by reference GAP and any other generic profile that describes a set of common requirements of the Bluetooth system.

6.4 GENERIC ATTRIBUTE ARCHITECTURE

6.4.1 Attribute Protocol

To allow devices to read and write small data values held on a server, an Attribute Protocol (ATT) is defined. Each stored value, typically only a few octets, is known as an attribute. This protocol allows each attribute to be self-identifying using UUIDs to identify the type of data. These UUIDs can be well-known assigned numbers defined in the Assigned Numbers document and associated specifications, or a vendor assigned 128-bit UUID.

Attribute Protocol messages are sent over L2CAP channels, known as the ATT bearers.

Attribute Protocol defines two roles: Client and Server. A device can be both an ATT Client and an ATT Server at the same time. Attribute Protocol messages on a single ATT bearer allow a single transaction in each direction to be outstanding at a time. When a response to a message is received, the next transaction can be initiated. When multiple ATT bearers are created, each ATT bearer has a separate transaction model and therefore multiple ATT transactions can be outstanding at the same time, one per bearer. This can be used to allow multiple higher layer specifications to send messages concurrently.

The ATT Server stores the attributes and accepts Attribute Protocol requests, commands and confirmations from the ATT Client. The ATT Server sends responses to requests and, when configured by a higher layer, sends indications and notifications asynchronously to the ATT Client when specified events occur on the ATT Server.

6.4.2 Generic Attribute Profile

Generic Attribute Profile (GATT) is built on top of the Attribute Protocol (ATT) and establishes common operations and a framework for the data transported and stored by the Attribute Protocol. GATT defines two roles: Server and Client. A GATT Client or Server is an ATT Client or Server respectively that conforms to the requirements in GATT. The GATT roles are not necessarily tied to specific GAP roles but may be specified by higher layer profiles. GATT and ATT are not transport specific and can be used in both BR/EDR and LE. However, GATT and ATT are mandatory to implement in LE since it is used for discovering services.

GATT also specifies the format of data contained on the GATT Server. Attributes, as transported by the Attribute Protocol, are formatted as Services and Characteristics. Services may contain a collection of characteristics. Characteristics contain a single value and any number of descriptors describing the characteristic value.

With the defined structure of services, characteristics and characteristic descriptors a GATT Client that is not specific to a profile can still traverse the GATT Server and display characteristic values to the user. The characteristic descriptors can be used to display descriptions of the characteristic values that may make the value understandable by the user.

6.5 GATT-BASED PROFILE HIERARCHY

The GATT Profile specifies the structure in which profile data is exchanged. This structure defines basic elements such as services and characteristics, used in a profile.

The top level of the hierarchy is a profile. A profile is composed of one or more services necessary to fulfill a use case. A service is composed of characteristics or references to other services. Each characteristic contains a value and may contain optional information about the value. The service and characteristic and the components of the characteristic (i.e., value and descriptors) contain the profile data and are all stored in Attributes on the server.

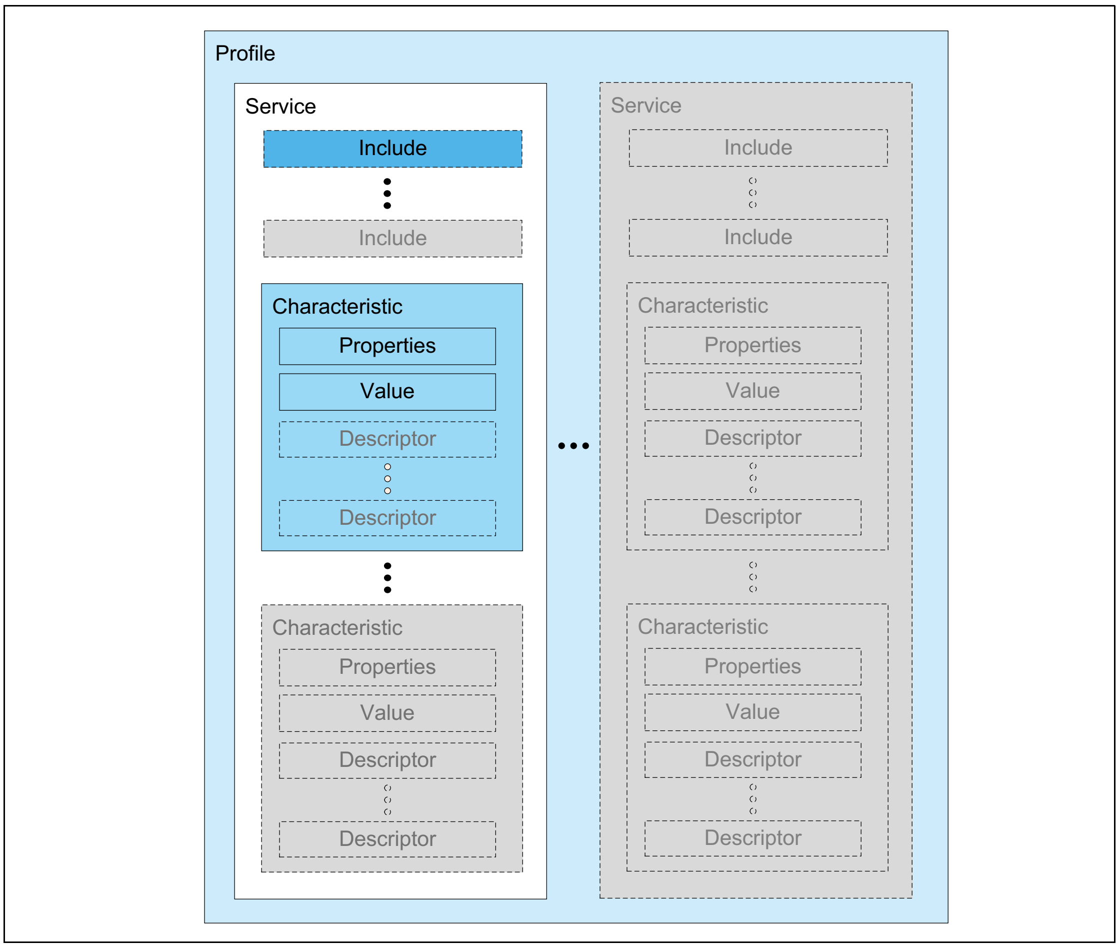

Figure 6.3: GATT-Based Profile hierarchy

6.5.1 Service

A service is a collection of data and associated behaviors to accomplish a particular function or feature of a device or portions of a device. A service may include other primary or secondary services and/or a set of characteristics that make up the service.

There are two types of services: primary and secondary. A primary service is a service that provides functionality of a device that can be used on its own. A secondary service is a service that provides additional functionality of a device in association with a primary service and is included from at least one primary service on the device.

To maintain backward compatibility with earlier clients, later revisions of a service definition can only add new included services or optional characteristics. Later revisions of a service definition are also forbidden from changing behaviors from previous revision of the service definition.

Services may be used in one or more profiles to fulfill a particular use case.

6.5.2 Included services

An included service is a method to incorporate another service definition on the server as part of the service including it. When a service includes another service, the entire included service becomes part of the new service including any nested included services and characteristics. The included service still exists as an independent service. There are no limits to the depth of nesting.

6.5.3 Characteristic

A characteristic is a value used in a service along with properties and configuration information about how the value is accessed and information about how the value is displayed or represented. A characteristic definition contains a characteristic declaration, characteristic properties, and a value. It may also contain descriptors that describe the value or permit configuration of the server with respect to the characteristic value.

6.6 MESH-BASED MODEL HIERARCHY

The Mesh Profile[1] specifies the structure within which data is exchanged by devices in a mesh network. This structure defines basic building blocks such as models and properties.

The top level of the hierarchy is a model, which is either a client model or a server model. A client model can send messages to a server model and the server model can use other messages to respond to the client model. Models enable devices to send standardized messages, using standardized data formats, to other devices that they have had no previous relationship with.

6.6.1 Model

A model is a collection of properties, states, messages, and associated behaviors that accomplishes a particular device function. A model defines and exposes states along with any associated behavior. It also defines the messages that are used to communicate between models within devices in a mesh network. Messages are defined globally and are not model-specific. Models are immutable meaning that features cannot be added to or removed from a model definition. Therefore, the only way to add features to a model is by defining a new model that extends an existing model by defining new states, messages, and behaviors. These new states and behaviors are linked to the existing model using state binding. This ensures backwards and forwards compatibility by allowing newer devices to access the newer features of the extended model, and older devices to access the base features of an existing model.

1. The Mesh Profile specification is available at https://www.bluetooth.com/specifications.

6.6.2 Properties

A property adds context to a defined characteristic. When sending data into a mesh network, it is very useful to label data with the meaning, or context, of that data. This allows a device that receives a property to interpret that data without having to negotiate the context beforehand. For example, a temperature characteristic can be given a context about how or when that temperature was measured such as “outside temperature”, “indoor temperature”, or “oil temperature”.

Properties are defined globally and are not model-specific.