GATT based Profile Specifications

- Profile specifications define
  - Use case
  - Behaviors
  - Discovery Procedures
  - Connection Parameters (slave latency, conn Interval) etc
  - Profile Roles
- Service specifications define
  - Characteristics (Mandatory, Optional)
  - Characteristics Properties (Broadcast, Control Point etc)
- Characteristics specifications define
  - Specify structure of value – Eg: Alert Level – 1 byte
  - Permissible values – Eg: 0 – No Alert, 1 – Medium Alert, 2 – High Alert
  - Permissions – Read/Write

2011 Published Profiles

- Alert Notification Profile
- Alert Notification Service
- Blood Pressure Profile
- Blood Pressure Service
- Current Time Service
- Device Information Service
- Find Me Profile
- Health Thermometer Profile
- Health Thermometer Service
- Heart Rate Profile
- Heart Rate Service
- Immediate Alert Service
- Link Loss Service
- Next DST Change Service
- Phone Alert Status Profile
- Phone Alert Status Service
- Proximity Profile
- Reference Time Update Svc
- Time Profile
- Tx Power Service

Adopted Profiles Example

Example: Heart Rate Profile

- User Scenarios
  - The Heart Rate Profile is used to enable a data collection device to obtain data from a Heart Rate Sensor that exposes the Heart Rate Service
- Roles
  - Heart Rate Sensor
  - Heart Rate Collector
- Heart Rate Sensor Role requirements
  - Heart Rate Service - Mandatory
  - Device Information Service - Manadatory
- Characteristics – Heart Rate Service
  - Heart Rate Measurement - Notify .
  - Heart Rate Measurement - Client Characteristic Configuration descriptor
  - Body Sensor Location - Read - Optional

Example: Proximity Profile

- User Scenarios
  - Leaving a phone behind
  - Leaving keys behind
  - Child straying too far
  - Hospital patient from bed
  - Automatic PC Locking & Unlocking
  - Automatic PC Locking & Authenticated Unlocking
- Roles
  - Proximity Monitor
  - Proximity Reporter
- Proximity Profile
  - Specifies services used
  - Specifies GAP requirements for discoverability/connectability
- Services
  - Link Loss Service
  - Immediate Alert Service
  - Tx Power Service

UUIDs for Adopted Profiles

- Gatt framework allows defining custom profiles
- Bluetooth_Base_UUID: **00000000-0000-1000-8000 00805F9B34FB**.
- 16bit in red reserved for Adopted Profiles
- There are no UUIDs for Profiles. They are only defined forServices , Characteristics , Descriptors and Declarations
- Example Adopted Profile – Proximity Profile:
  - Link Loss Service UUID: 0x1803
  - Immediate Alert Service UUID: 0x1802
  - TX power Service UUID; 0x1804

Custom Profile Example – Hello Bluetooth

UUIDs for Custom Profiles

- Custom Profile UUID – Need to generate 128 bit UUID
  - Refer to The ITU-T Rec. X.667. You can download a free copy of ITU-T Rec. X.667 from http://www.itu.int/ITU-T/studygroups/com17/oid/X.667-E.pdf.
  - Generating a 128 bit UUID: http://www.itu.int/ITU-T/asn1/uuid.html, you can generate a unique 128-bit UUID.
- Hello Bluetooth Profile UUIDs (Asssuming we reserved 5ab2xxxx-b355-4d8a-96ef-2963812dd0b8)
  - Service: Hello Server - 5ab20001-b355-4d8a-96ef-2963812dd0b8
  - Characteristic: Username - 5ab20002-b355-4d8a-96ef-2963812dd0b8

Hello Bluetooth Profile

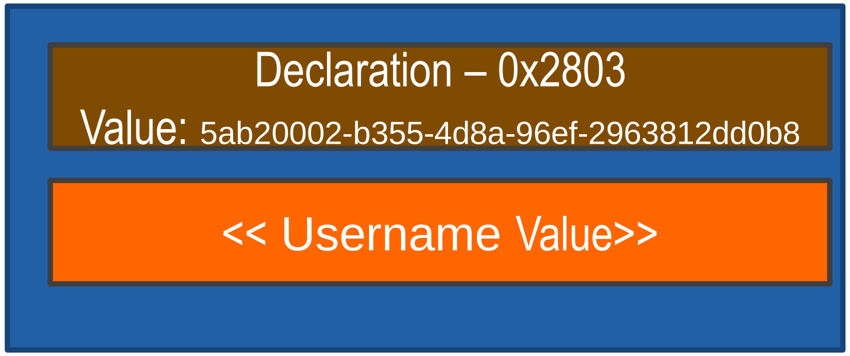
- Education Profile – Demonstrate creating custom profiles
- Creating a Custom Profile Process
  - Step 1: Articulating Use Case
  - Step 2: Identifying Characteristics
  - Step 3: Defining Services
  - Step 4: Defining Profile
  - Step 5: Generating Attribute Table
- Hello Bluetooth XML representation

STEP 1: Articulating a Use Case

- Bluetooth Enabled Business Cards
- Bluetooth Enabled Registration desk
- Use Case
  - Person walks to the registration desk.
  - Based on proximity (measured by RSSI), business card establishes a connection
  - Exchange information (Name)
  - “Welcome to <Confernece> 2012, Name”

Step 2: Identifying Characteristics

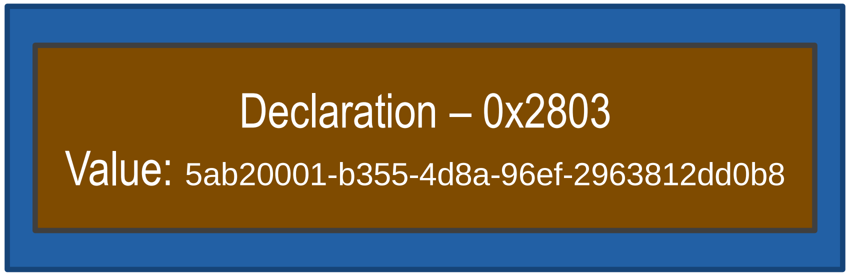
- Characteristics are defined attribute types that contain a single logical value.
- Characteristic: <<Username>>.
- UUID Generator:
  - 128 bit characteristics UUID
  - 5ab20002-b355-4d8a-96ef-2963812dd0b8
- Permissions: Read
- Size: utf-8 string



Step 3: Defining Service

- Services are collections of characteristics and relationships to other services that encapsulate the behavior of part of a device.
- Service: <<Hello Service>>
  - UUID Generator: 128 bit characteristics UUID

    5ab20001-b355-4d8a-96ef-2963812dd0b8
- Characteristic: <<Username>> - mandatory



Declaration – 0x2803

Value: 5ab20001-b355-4d8a-96ef-2963812dd0b8

Step 4: Profile

- Profiles are high level definitions that define how services can be used to enable an application or use case.
- Roles
  - Hello Server
  - Hello Client
- Connection Parameters
  - Connection Interval: 80 msec
  - Slave Latency: 0
  - Supervision Timeout: 2 seconds
- Hello Client Behaviors
  - If RSSI value > threshold, read <<name>> , display “Welcome to AHM, <<name>>
  - If RSSI value < threshold, clear the display

Attribute Table Example – Hello Server

Handle	Attribugte Type	Value	Permissi ons
0x00030	«Primary Service Declaration» 0x2800	«Hello Service» 5ab2d876-b355-4d8a-96ef-2963812dd0b8	R
0x00031	«Characteristic Declaration» 0x2803	{r, 0x0003, «User Name»}	R
0x00032	«User Name» 5a50528d-e5ba-4620-90ac-33e5b913684c	“Muhammad”	R

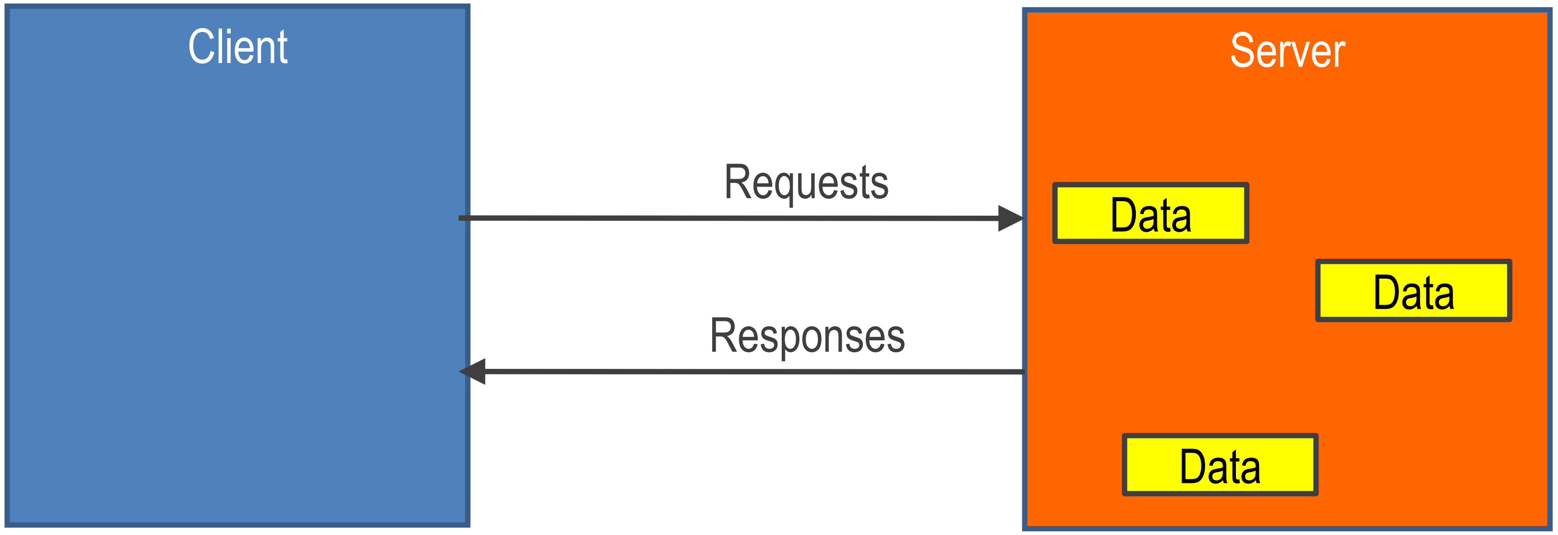
Hello Bluetooth XML representation

- GATT schema
  - http://schemas.bluetooth.org
  - http://schemas.bluetooth.org/Documents/profile.xsd
  - http://schemas.bluetooth.org/Documents/service.xsd
  - http://schemas.bluetooth.org/Documents/characteristic.xsd
- GATT based profiles are represented in xml files which follow the rules specified by the xsd files
  - HelloBluetoothProfile.xml
  - HelloService.xml
  - Username.xml

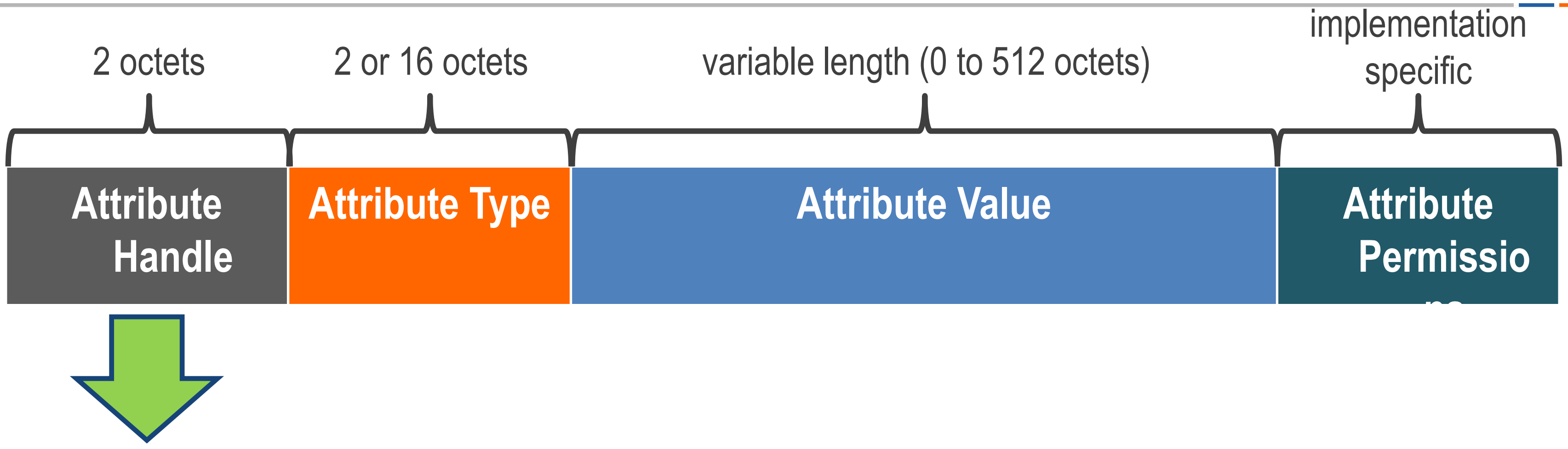
GATT Framework Background

Attribute Protocol (ATT)

- Client Server Architecture
  - servers have data
  - clients request data to/from servers
- Servers expose data using Attributes

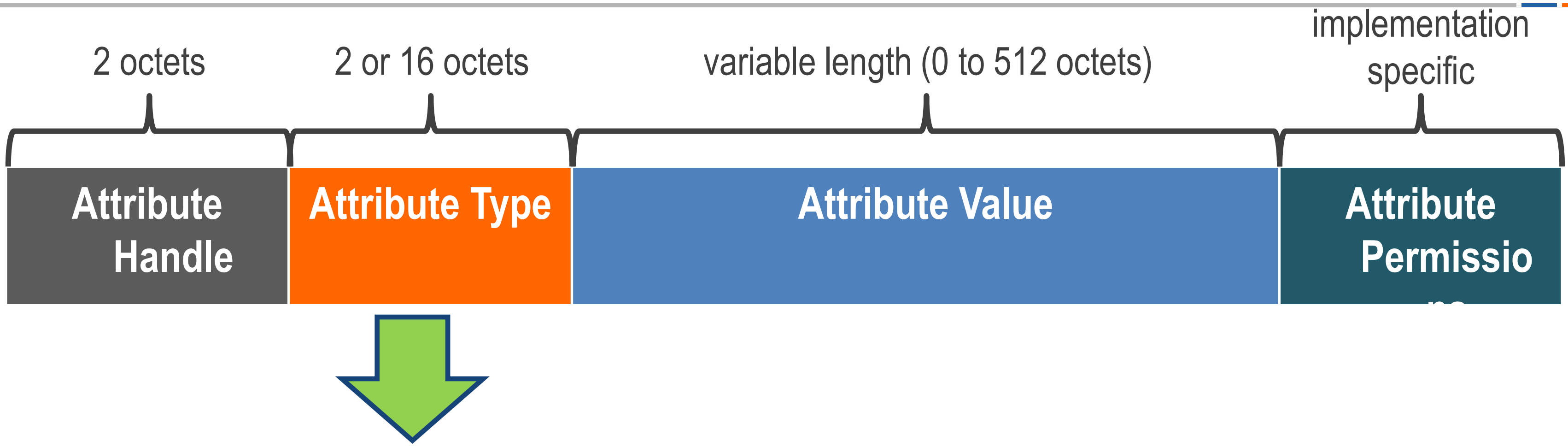


Attribute Handle



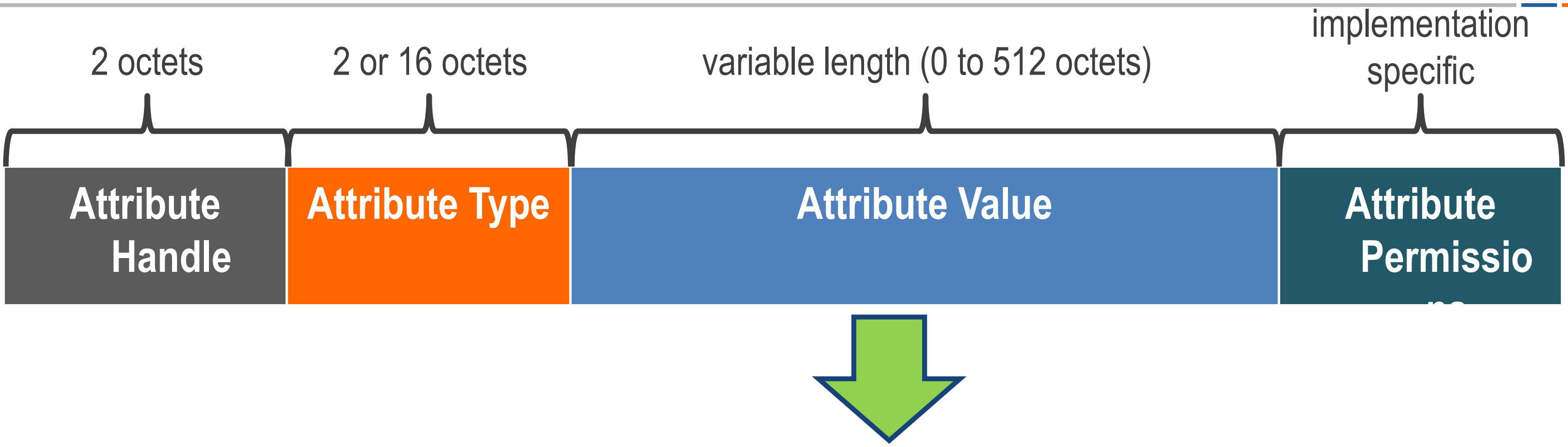
- Handle is a 16 bit value
  - 0x0000 is reserved – shall never be used
  - 0x0001 to 0xFFFF can be assigned to any attributes
- Handles are "sequential"
  - 0x0005 is "before" 0x0006
  - 0x0104 is "after" 0x00F8
- Always unique in the table

Attribute Type



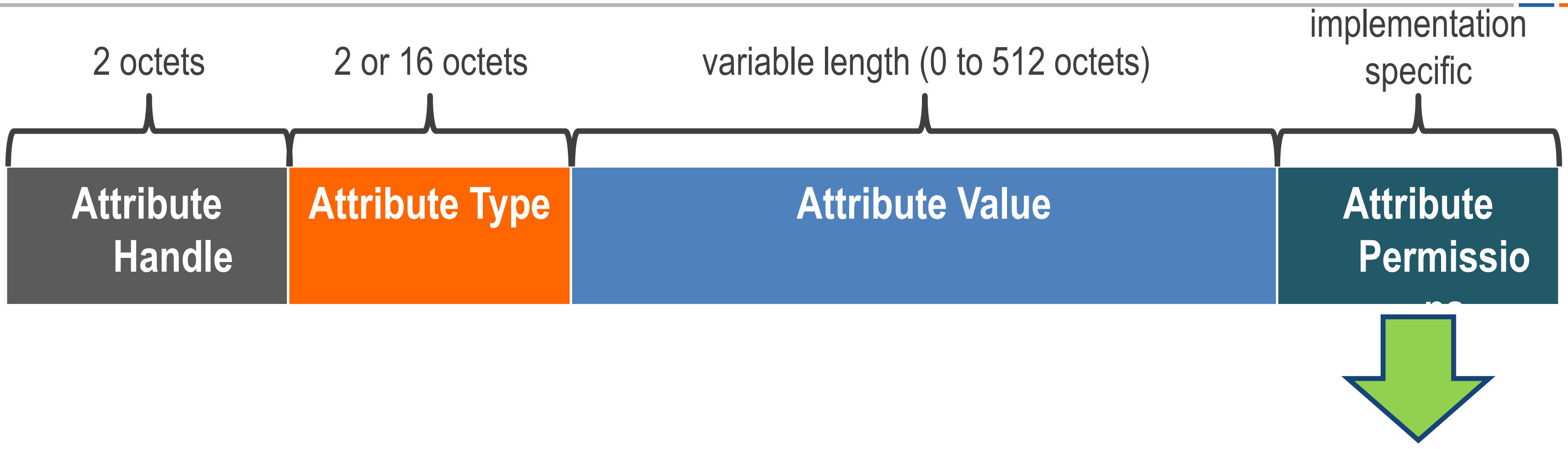
- SIG defined Attribute Types – 16 bits
  - Bluetooth_Base_UUID is: 00000000-0000-1000-8000 00805F9B34FB
  - Declarations - Defined GATT profile attribute types.
  - Descriptors - Defined attributes that describe a characteristic value
  - Numbers assigned to adopted services and characteristics
- Custom Attribute Types – 128 bit
  - Custom Services and characteristics
  - http://www.itu.int/ITU-T/asn1/uuid.html

Attribute Value



- An Attribute value is an array of octets, 0 to 512 octets in length can be fixed or variable length
- Each Attribute type defines the data structure for the attirbute value
  - Example: AttributeType = 0x2800 defines a 16 or 128 bit value
  - Example: Attribute Type = 0x2803 defines the Attribute Value to be {r, «Handle», «UUID»}
  - Example: Attribute Type = AlertLevel(0x2A06) defines Attribute value to be uint8

Attribute Permissions



- Attributes values may be:
  - readable / not readable
  - writeable / not writeable
  - readable & writeable / not readable & not writeable
- Attribute values may require:
  - authentication to read / write
  - authorization to read / write
  - encryption / pairing with sufficient strength to read / write

- Permissions not “discoverable” over Attribute Protocol
- If request to read an attribute value that cannot be read - Error Response «Read Not Permitted»
- If request to write an attribute value that requires authentication - Error Response «Insufficient Authentication» - Client must create authenticated connection and then retry
- There is no “pending” state

PROTOCOL METHODS

Protocol PDU Type	Sent by	Description
Request	Client	Client requests something from server – always causes a response
Response	Server	Server sends response to a request from a client
Command	Client	Client commands something to server – no response
Notification	Server	Server notifies client of new value – no confirmation
Indication	Server	Server indicates to client new value – always causes a confirmation
Confirmation	Client	Confirmation to an indication

PROTOCOL IS STATELESS

- After transcation complete
  - no state is stored in protocol

- A transaction is:
  - Request -> Response
  - Command
  - Notification
  - Indication -> Confirmation

SEQUENTIAL PROTOCOL

- Client can only send one request at a time
  - request completes after response received in client
- Server can only send one indication at a time
  - indication completes after confirmation received in server
- Commands and Notifications are no response / confirmation
  - can be sent at any time
  - could be dropped if buffer overflows – consider unreliable

Client Initiated Methods

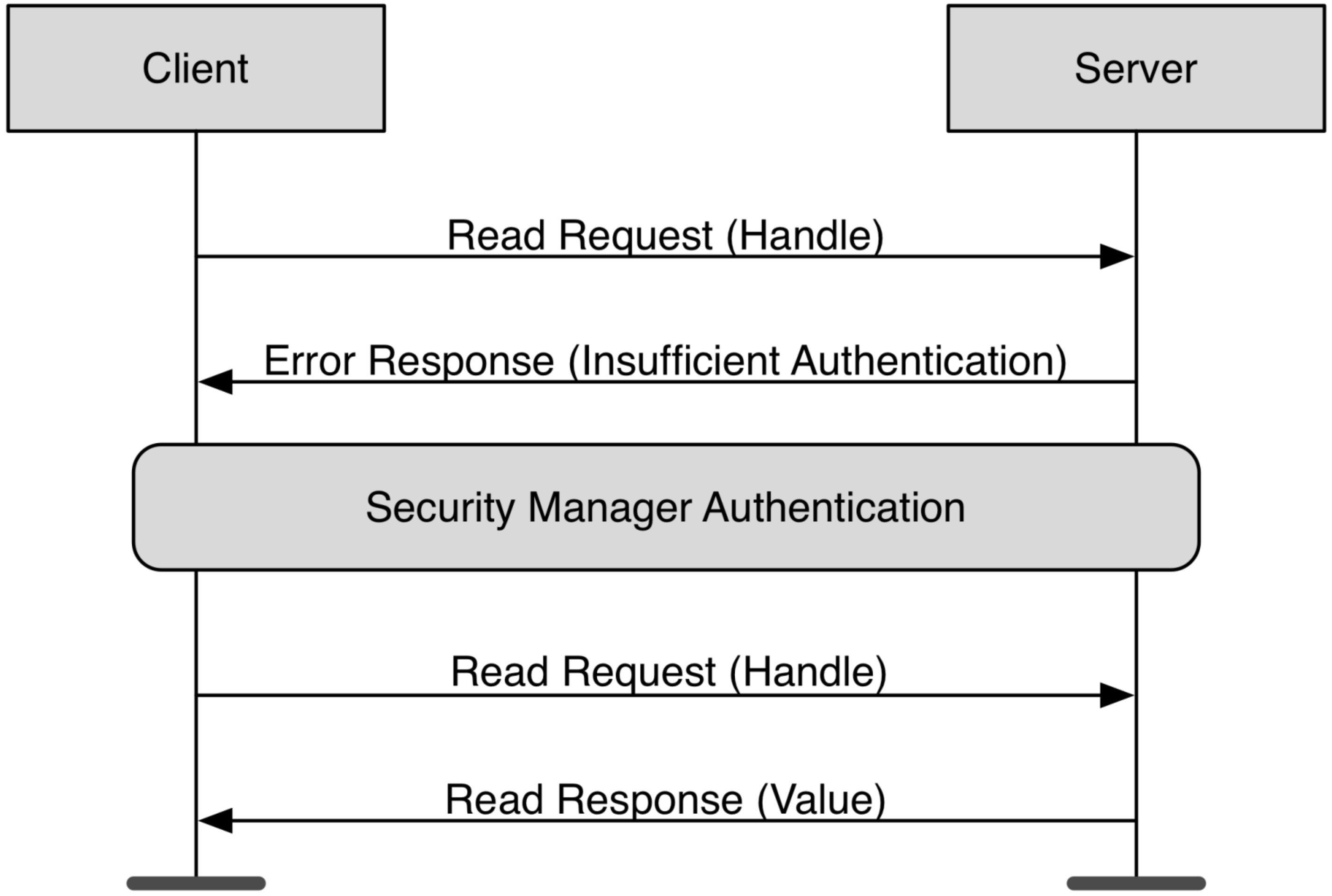
- Request Method
  - Reading Attributes
    - ReadRequest(handle) ←→ ReadResponse(value)
    - ReadByTypeRequest(startingHandle, endHandle, UUID) ←→ ReadByTypeResponse(list of [handle, value] pair)
  - Writing Attributes
    - WriteRequest(handle, value) ←→ WriteReponse
  - Finding Attributes
    - FindInformation(startingHandle, endHandle, UUID) ←→ FindInformationResponse(format, [Handle, UUID])
- Example
  - Read (0x0022) => 0x04 ; Read (0x0098) => 0x0802

Server Initiated Methods

- `Handle Value Notification (handle, value)`
- `Handle Value Indication(handle, value) => Handle Value Confirmation ()`

Error Response

(any) Request (*) => Error Response (Opcode, Handle, Error Code)

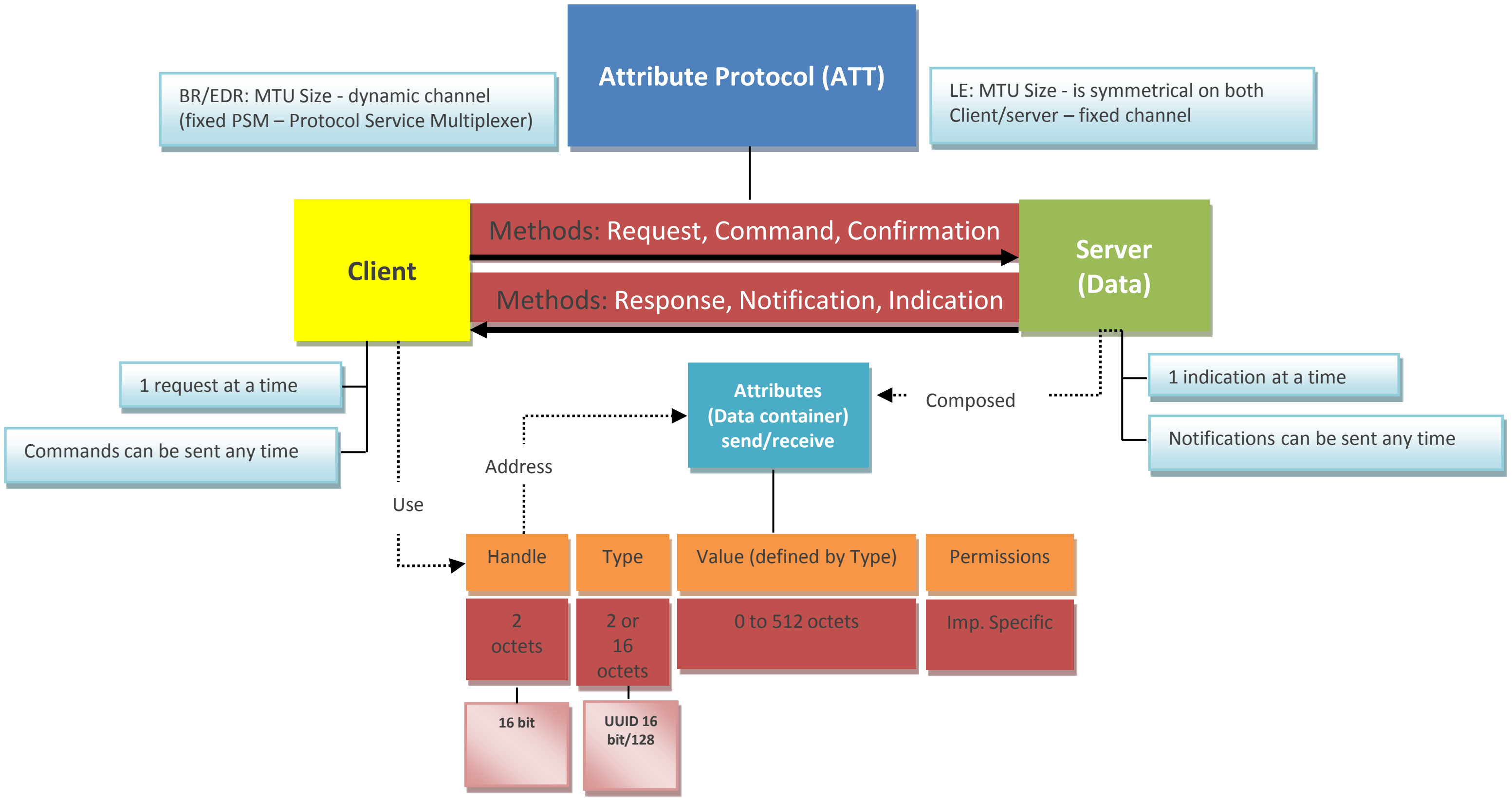

Client
Server
Read Request (Handle)
Error Response (Insufficient Authentication)
Security Manager Authentication
Read Request (Handle)
Read Response (Value)

Example: ATTRIBUTE Table

- Example – ReadRequest(0x0022) - ReadResponse(0x802)
- Example – ReadRequest(0x0004) – ReadResposne({r, 0x0006, <<Appearance>>

Handle	Type	Value	Permissions
0x0001	«Primary Service»	«GAP»	R
0x0002	«Characteristic»	{r, 0x0003, «Device Name»}	R
0x0003	«Device Name»	“Temperature Sensor”	R
0x0004	«Characteristic»	{r, 0x0006, «Appearance»}	R
0x0006	«Appearance»	«Thermometer»	R
0x000F	«Primary Service»	«GATT»	R
0x0010	«Characteristic»	{r, 0x0012, «Attribute Opcodes Supported»}	R
0x0012	«Attribute Opcodes Supported»	0x00003FDF	R
0x0020	«Primary Service»	«Temperature»	R
0x0021	«Characteristic»	{r, 0x0022, «Temperature Celsius»}	R
0x0022	«Temperature Celsius»	0x0802	R*

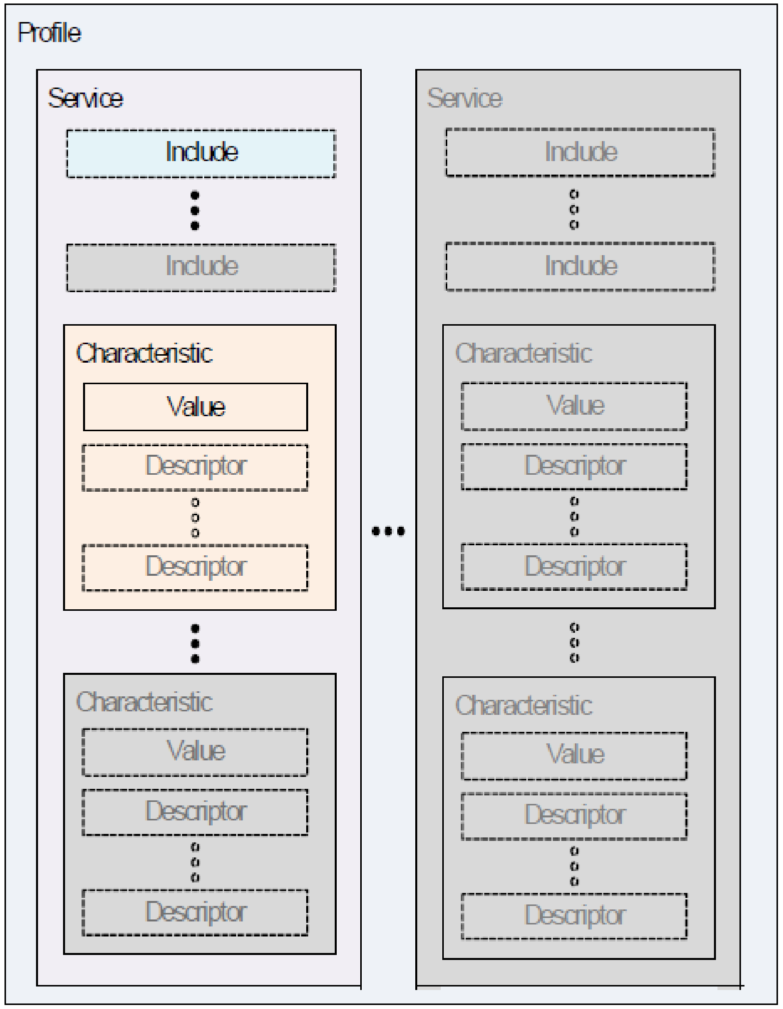
Attribute Protocol (ATT) Summary


Attribute Protocol (ATT)
BR/EDR: MTU Size - dynamic channel (fixed PSM – Protocol Service Multiplexer)
LE: MTU Size - is symmetrical on both Client/server – fixed channel
Client
Methods: Request, Command, Confirmation
Methods: Response, Notification, Indication
Server (Data)
1 request at a time
Commands can be sent any time
Attributes (Data container) send/receive
Composed
1 indication at a time
Notifications can be sent any time
Address
Use
Handle
Type
Value (defined by Type)
Permissions
2 octets
2 or 16 octets
0 to 512 octets
Imp. Specific
16 bit
UUID 16 bit/128

- Exposes Data using Typed, Addressable Attributes: Handle, Type, Value
- Methods for finding, reading, writing attributes by client
- Methods for sending notifications / indications by server

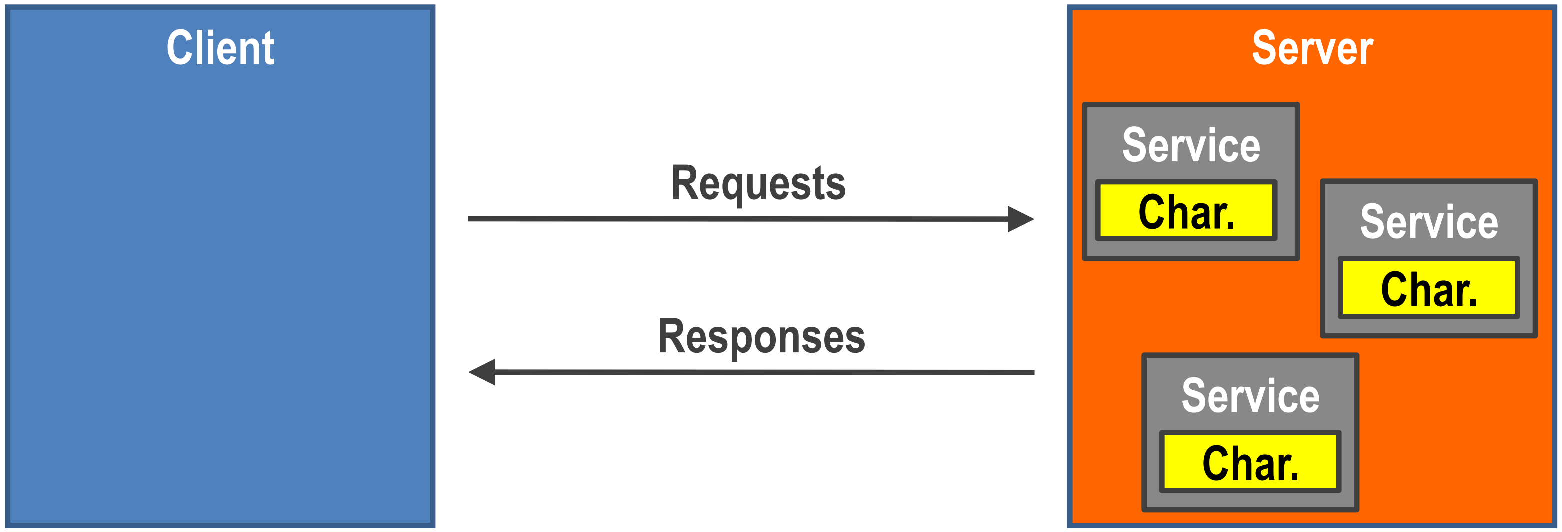
GENERIC ATTRIBUTE PROFILE (GATT) Hierarchy

- Built on top of the ATT
- Provides a framework for developing profiles
- A profile is composed of one or more services.
- A service is composed of characteristics or references to other services.
- Each characteristic contains a value and may contain optional information about the value.



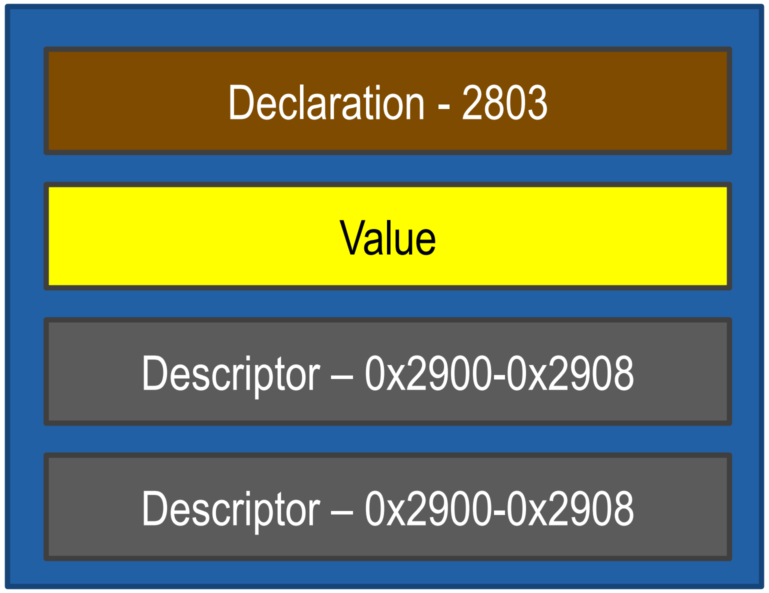
Client Server Architecture

- Same client server architecture as Attribute Protocol
  - except that data is encapsulated in “Services”
  - data is exposed in “Characteristic”



WHAT IS A CHARACTERISTIC?

- Group of attributes to define data
- Characteristics specify
  - Data size, format
  - Permissible Values
  - Permissions
- Represented in Attribute Table as multiple attirbutes
  - Characteristic Declaration
  - Characteristic Value
  - Characteristic Descriptors – 1 : n
- Example – Alert Level
  - Uint8
  - Permissbile values: 0, 1, 2
  - R/W



ATTRIBUTES ARE FLAT

Handle	Type	Value	Permissions
0x0001	«Primary Service»	«GAP»	R
0x0002	«Characteristic»	{r, 0x0003, «Device Name»}	R
0x0003	«Device Name»	“Temperature Sensor”	R
0x0004	«Characteristic»	{r, 0x0006, «Appearance»}	R
0x0006	«Appearance»	«Thermometer»	R
0x000F	«Primary Service»	«GATT»	R
0x0010	«Characteristic»	{r, 0x0012, «Attribute Opcodes Supported»}	R
0x0012	«Attribute Opcodes Supported»	0x00003FDF	R
0x0020	«Primary Service»	«Temperature»	R
0x0021	«Characteristic»	{r, 0x0022, «Temperature Celsius»}	R
0x0022	«Temperature Celsius»	0x0802	R*

GROUPING GIVES STRUCTURE

Handle	Type	Value	Permissions
0x0001	«Primary Service»	«GAP»	R
0x0002	«Characteristic»	{r, 0x0003, «Device Name»}	R
0x0003	«Device Name»	"Temperature Sensor"	R
0x0004	«Characteristic»	{r, 0x0006, «Appearance»}	R
0x0006	«Appearance»	«Thermometer»	R
0x000F	«Primary Service»	«GATT»	R
0x0010	«Characteristic»	{r, 0x0012, «Attribute Opcodes Supported»}	R
0x0012	«Attribute Opcodes Supported»	0x00003FDF	R
0x0020	«Primary Service»	«Temperature»	R
0x0021	«Characteristic»	{r, 0x0022, «Temperature Celsius»}	R
0x0022	«Temperature Celsius»	0x0802	R*

GROUPING GIVES STRUCTURE

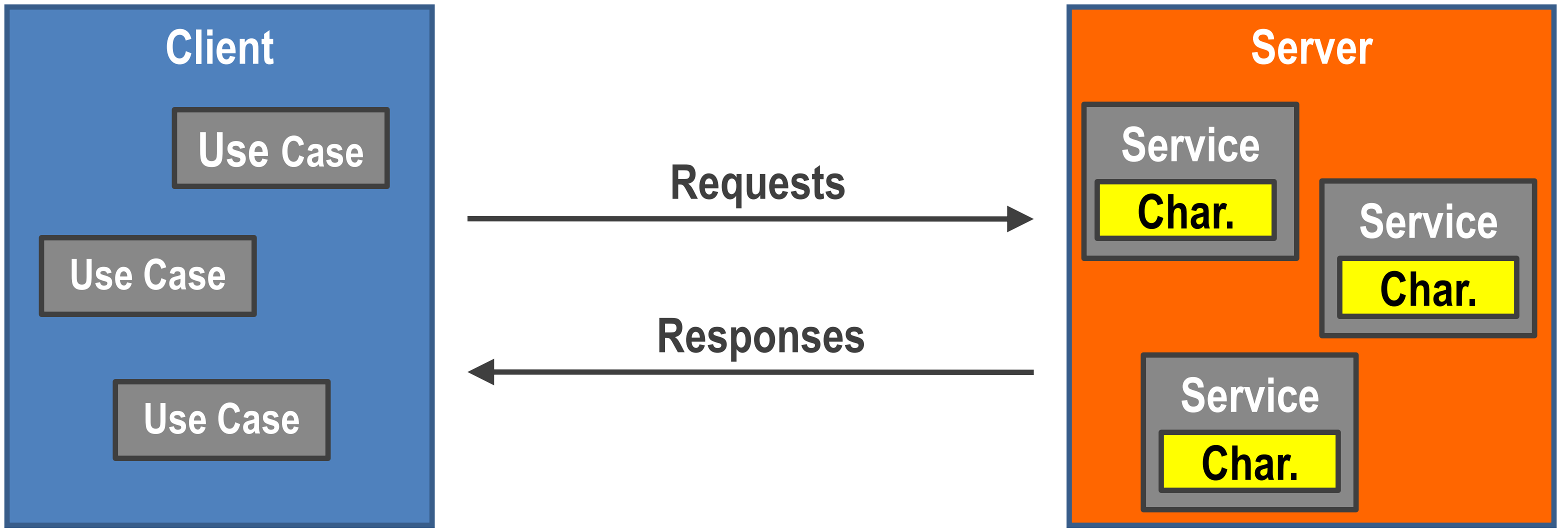
Handle	Type	Value	Permissions
0x0001	«Primary Service»	«GAP»	R
0x0002	«Characteristic»	{r, 0x0003, «Device Name»}	R
0x0003	«Device Name»	“Temperature Sensor”	R
0x0004	«Characteristic»	{r, 0x0006, «Appearance»}	R
0x0006	«Appearance»	«Thermometer»	R
0x000F	«Primary Service»	«GATT»	R
0x0010	«Characteristic»	{r, 0x0012, «Attribute Opcodes Supported»}	R
0x0012	«Attribute Opcodes Supported»	0x00003FDF	R
0x0020	«Primary Service»	«Temperature»	R
0x0021	«Characteristic»	{r, 0x0022, «Temperature Celsius»}	R
0x0022	«Temperature Celsius»	0x0802	R*

GROUPING GIVES STRUCTURE

Handle	Type	Value	Permissions
0x0001	«Primary Service»	«GAP»	R
0x0002	«Characteristic»	{r, 0x0003, «Device Name»}	R
0x0003	«Device Name»	"Temperature Sensor"	R
0x0004	«Characteristic»	{r, 0x0006, «Appearance»}	R
0x0006	«Appearance»	«Thermometer»	R
0x000F	«Primary Service»	«GATT»	R
0x0010	«Characteristic»	{r, 0x0012, «Attribute Opcodes Supported»}	R
0x0012	«Attribute Opcodes Supported»	0x00003FDF	R
0x0020	«Primary Service»	«Temperature»	R
0x0021	«Characteristic»	{r, 0x0022, «Temperature Celsius»}	R
0x0022	«Temperature Celsius»	0x0802	R*

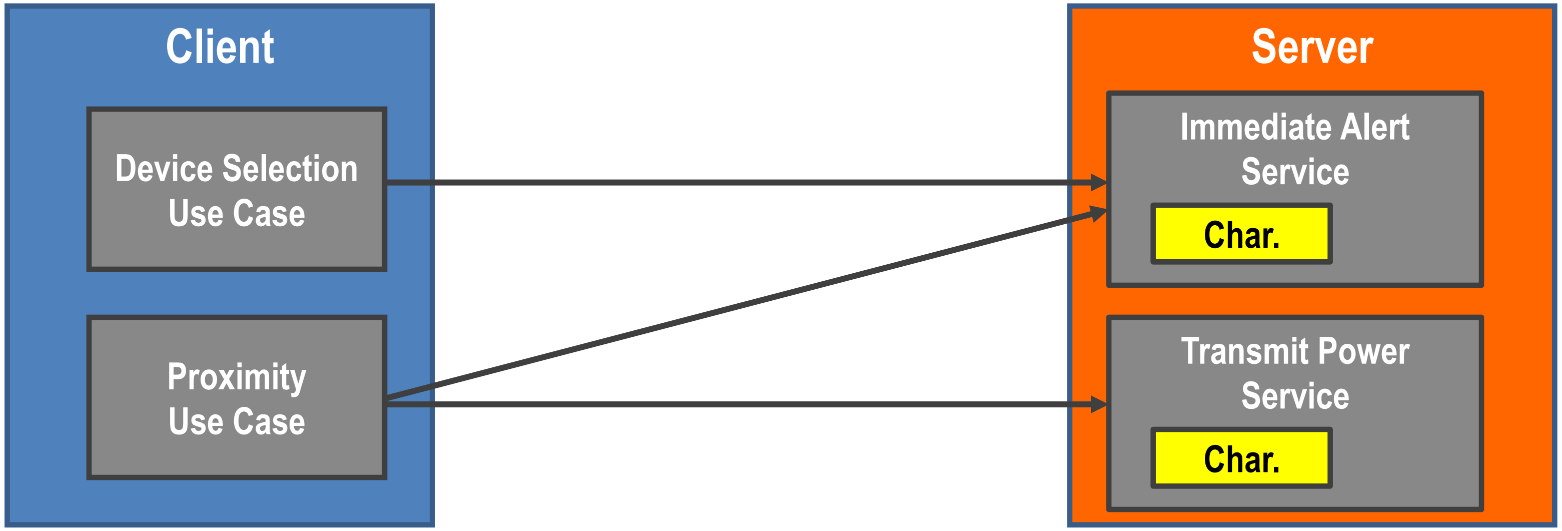
GATT – Generic Attribute Protocol

- Client Server Architecture
  - Services – exposes behavior that have characteristics
  - Use Cases– define how to use services on a peer



Use Cases and Services

- There is not a one-to-one link between services and use cases
- Clients implement use cases, Servers implement services
- Use cases can use multiple services


Client
Device Selection Use Case
Proximity Use Case
Server
Immediate Alert Service
Char.
Transmit Power Service
Char.