# ECEN 5823-001
# Internet of Things Embedded Firmware

Lecture #14

11 October 2018

Be Boulder.

University of Colorado Boulder

# Agenda

- Class Announcements
- Mid-term structure
- Bluetooth Low Energy / Smart

# Class Announcements

- Quiz #7 is due at 11:59 on Sunday, October 21th, 2018
- ~~Homework #4: BLE Server + MITM + LCD due on Sunday, October 7th, at 11:59pm~~
- ~~Homework #5: Client flowchart due on Wednesday, October 10th, at 11:59pm~~
- Homework #6: Client-Server due on Wednesday, October 17th, at 11:59pm
- Mid-term in class on Thursday the 18th
  - Attendance is required for on-campus students

# Mid-term structure

- Attendance is a must for on-campus students – Thursday the 18th
  - A zero will be recorded for on-campus students that do not take it in class
  - Be on-time!
- CU Honor Code is in affect
- A single sheet of 8x11 paper, both sides, of notes is allowed
  - These notes must be of your own
  - They will be handed in at the end of the mid-term
  - If the notes are copied or determined not individual work, a zero will be recorded for the mid-term and a CU Honor code violation will be initiated resulting in a “F” in the course
- Distant students will have until 11:59pm on Friday the 19$^{th}$ to complete the exam

# Mid-term structure

- Mid-term will comprise of 60 questions taken via Canvas
  - 40 questions from the Quiz 1 thru 6 question banks
  - 20 questions from a mid-term question bank
- If there is a canvas question issue such as no way to input an answer, please submit the issue to me after the exam via slack.
  - I will make a manual adjustment
- You can ask for clarification from the mid-term proctor

# BLE: Attributes

- Attribute database example
  - GAP Service Attribute
  - TX Power Service
  - Battery Service

| Attribute Handle | Attribute Type | Attribute Value |
|---|---|---|
| 0x0001 | Primary Service | GAP Service |
| 0x0002 | Characteristic | Device Name |
| 0x0003 | Device Name | "Proximity Tag" |
| 0x0004 | Characteristic | Appearance |
| 0x0005 | Appearance | Tag |
| 0x0006 | Primary Service | GATT Service |
| 0x0007 | Primary Service | Tx Power Service |
| 0x0008 | Characteristic | Tx Power |
| 0x0009 | Tx Power | –4dBm |
| 0x000A | Primary Service | Immediate Alert Service |
| 0x000B | Characteristic | Alert Level |
| 0x000C | Alert Level | |
| 0x000D | Primary Service | Link Loss Service |
| 0x000E | Characteristic | Alert Level |
| 0x000F | Alert Level | "high" |
| 0x0010 | Primary Service | Battery Service |
| 0x0011 | Characteristic | Battery Level |
| 0x0012 | Battery Level | 75% |
| 0x0013 | Characteristic Presentation Format | uint8, 0, percent |
| 0x0014 | Characteristic | Battery Level State |
| 0x0015 | Battery Level State | 75%, discharging |
| 0x0016 | Client Characteristic Configuration | 0x0001 |

**Figure 10–11** An example of an attribute database

# Silicon Labs software creates these attributes

Characteristic
Name: Glucose Measurement
Type: org.bluetooth.characteristic.glucose_measurement
UUID: 2A18
Summary:
The Glucose Measurement characteristic is a variable length structure containing a Flags field, a Sequence Number field, a Base Time field and, based upon the contents of the Flags field, may contain a Time Offset field, Glucose Concentration field, Type-Sample Location field and a Sensor Status Annunciation field.

Value:
Length: 19 byte
Variable length: false
Type: HEX

Property requirements:
Read - Excluded - false
Indicate - Excluded - false
Reliable write - Excluded - false
Write - Excluded - false
Write Without Response - Excluded - false
Notify - Mandatory - true
Descriptors:

*General settings*
Name: Glucose Measurement
User description

*Characteristic settings*
ID: glucose_measurement
UUID: 2A18
SIG type: org.bluetooth.characteristi

org.bluetooth.characteristic.glucose_measurement

*Value settings*
Value
Value type: hex
Length: 19 byte
Variable length

*Properties*
Set properties' information

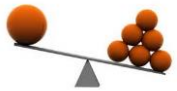
| Name | Requirement | State |
|---|---|---|
| Read | Excluded | False |
| Indicate | Excluded | False |
| Reliable write | Excluded | False |

*Capabilities*
Characteristic capabilities

# BLE: Attributes

- Some attributes contain information that can only be read or written
- To support these restrictions upon access, each and every attribute in an attribute database has a permission
  - Access permission
    - Determines what types of requests are permitted:  Readable, Writeable, or Readable/Writeable
    - If a request is attempted without the proper access permission, an error will be returned stating that the client cannot read this attribute or write to this attribute
  - Authentication permission
    - Determines whether Authentication is required or no authentication is required
    - Upon access to an attribute value, in addition to checking whether proper access permission, the attribute server checks with authentication is required.  If authentication is required, the client that sent the request is authenticated with the device
    - If the client has not been authenticated, an error stating that there is insufficient authentication will be returned

# BLE: Attributes



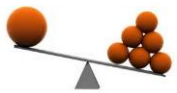
- If the client receives the insufficient authentication error, it can do one of two things:
  - Ignore the request and pass the error up to the application
  - Or, attempt to authenticate by using its Security Manager and resend the request
- The complexity of authentication and reissuing the request is up to the client, and the server is not required to hold state of the request

- Authorization permission
  - Upon access to an attribute value, in addition to checking whether proper access permission, the attribute server checks whether the client has been authorized. If authorization is required, the client that sent the request is determined whether it has been authorized
  - If the client has not been authorized and receives the error of insufficient authorization, it is an error that the client can not resolve
  - Authorization is a property of the server, and the server either authorizes the client or it does not

- Note: Attribute permissions only pertain to the Attribute value

# BLE: Attributes (accessing)

- Each attribute in the attribute database can be accessed using the following types of messages:
  - Find Requests: a client can find attributes in an attribute database so that the more efficient handle-based requests can be used
  - Read Requests: A request to read one or more attribute values using one or more handles or a range of handles to read
  - Write Requests: These requests always use an attribute handle and the value to write
  - These three requests always cause the attribute server to send a single response. If more data is required than what can fit in a single response, the client will need to resend the request by adjusting the attribute handles
  - To minimize complexity of the server, only one request can be sent at a time, and another request can only be sent after the previous response has been received



# BLE: Attributes (accessing)

- Each attribute in the attribute database can be accessed using the following types of messages:
  - Write Command does not require a response which makes it different than the write request
  - The notification which is sent by the server to the client without a request from the client does not require a response as well
    - Since the Write Command and Notifications do not require a response, they are considered unreliable, but they can be very useful in the correct use case
  - The last way to access an attribute is the Indication. An indication is similar to the notification in that the server provides this information without the request of the client, but an indication does require a response, so it is considered reliable

# BLE: Attributes (Prepare Write Requests and Execute Write Requests)

- These messages enable a device to prepare a whole sequence of writes and them executes them as a single transaction
- Each Prepare Write Request include:
  - The handle of the attribute to be written
  - The value of the attribute
  - An offset into the attribute's value where this value will be written
- Enabling each Prepare Write Request to write a portion of a large attribute value
- Prepare Write Response include:
  - The handle of the attribute to be written
  - The value of the attribute
  - An offset into the attribute's value where this value will be written
- The Prepare Write Response enables the client to verify that the server has the correct write request before issuing the Execute Write Request
- If the Prepare Write Response did not match the Prepare Write Request, the client to issue the "cancel" code in the Execute Write Request

# BLE: Attributes

- How will the BLE server respond to a Find Request if it has more data than a single response?
- What will the BLE client must do to obtain all the data that it is requesting?

| Attribute Handle | Attribute Type | Attribute Value |
|---|---|---|
| 0x0001 | Primary Service | GAP Service |
| 0x0002 | Characteristic | Device Name |
| 0x0003 | Device Name | "Proximity Tag" |
| 0x0004 | Characteristic | Appearance |
| 0x0005 | Appearance | Tag |
| 0x0006 | Primary Service | GATT Service |
| 0x0007 | Primary Service | Tx Power Service |
| 0x0008 | Characteristic | Tx Power |
| 0x0009 | Tx Power | –4dBm |
| 0x000A | Primary Service | Immediate Alert Service |
| 0x000B | Characteristic | Alert Level |
| 0x000C | Alert Level | |
| 0x000D | Primary Service | Link Loss Service |
| 0x000E | Characteristic | Alert Level |
| 0x000F | Alert Level | "high" |
| 0x0010 | Primary Service | Battery Service |
| 0x0011 | Characteristic | Battery Level |
| 0x0012 | Battery Level | 75% |
| 0x0013 | Characteristic Presentation Format | uint8, 0, percent |
| 0x0014 | Characteristic | Battery Level State |
| 0x0015 | Battery Level State | 75%, discharging |
| 0x0016 | Client Characteristic Configuration | 0x0001 |

**Figure 10–11** An example of an attribute database

# BLE Attributes

- If the Client wants to write a very long descriptor into the BLE server attribute database, what is the procedure?
- Is the write to the BLE server attribute an atomic operation?
- Why would you like this operation to be atomic?

# BLE: Attributes (Grouping)

- The Generic Attribute Protocol only defines a flat structure of attributes
  - Each Attribute has its handle as an address
- The Attribute Protocol defines groups as in object oriented programming of attributes
- Definitions:
  - An interface is a description of external behavior
  - A class is an implementation of that interface
  - An object is an instantiation of that class

# BLE: Attributes (Grouping)

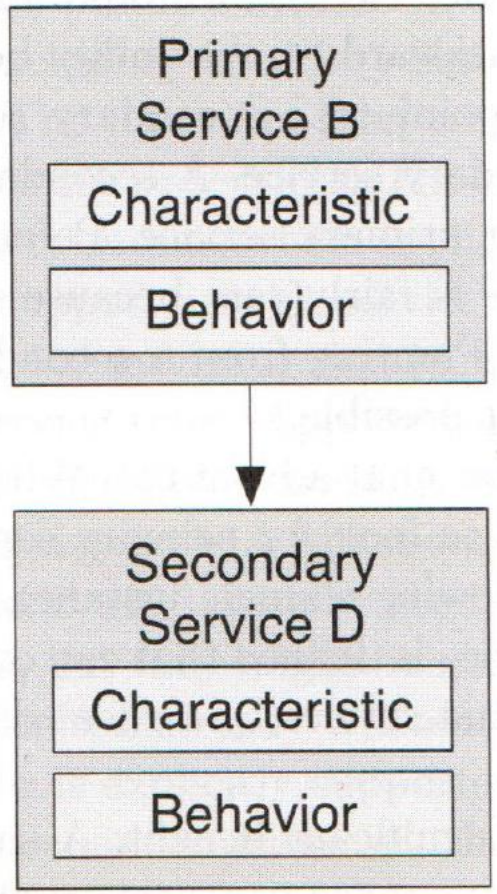
- Example:
  - A car is an instance of an automobile class that implements the driving interface
  - Not all car objects look the same, they can be implemented differently, but critically, they all have the same basic driving interface
    - Driving wheel, the accelerator pedal, and brake pedal
  - The driving interface is the same, but the class that implements this interface can be different
    - General Motors, BMWs, Hondas, etc.
  - A service is grouped by using a service declaration
    - Grouping of one or more characteristics
  - And, a characteristic is grouped using a characteristic declaration
    - Grouping of one or more attributes

# BLE: Services

- The Generic Attribute Profile defines two basic forms of groupings:
  - Services
  - And, Characteristics
- Service
  - Equivalent of an object that has immutable interface
  - Includes one or more characteristics
  - Can reference other services
  - The set of characteristics and their associated behavior encompasses the immutable interface for the service
- Characteristic
  - A unit of data or exposed behavior

# BLE: Services

- Primary and secondary services
  - A primary service exposes what a device typically does
    - Example: Battery gas gauge's primary service is to provide how much charge is in the battery
  - A device can have both a primary and secondary services
    - If a devices supports a Service B, Service B would be instantiated as a primary service
    - If the device has additional information, but not associated with the what the device does, this secondary service, Service D, would be instantiated as a secondary service.
  - A secondary service is an encapsulation of behavior and characteristics that are not something that a user would necessarily need to understand
    - Example: The battery gas gauge providing the temperature of the battery



# BLE: Services

- Secondary services can only be found by reference and must always have another service that points to them
- A primary service can have more than one secondary service which can create a tree of services
- Primary services can easily be located by a client that is looking for a particular service
  - This enables a simple client to locate whether a desired service is on a device with minimal effort
- Once the client builds up the “tree” of services or multiple “trees,” a “forest,” the client can pass this information to the application to determine how to use the services

# BLE: Services

- Service Declaration
  - A service is grouped by using a service declaration which is an attribute type of Primary Service or Secondary Service
  - Simple clients are devices that have no user interface but can still use services on a peer device
    - A simple device can just search for the primary services and find the services that it requires
    - Since the Primary services are directly accessible, the simple client does not need to walk through the “tree” of services to locate it

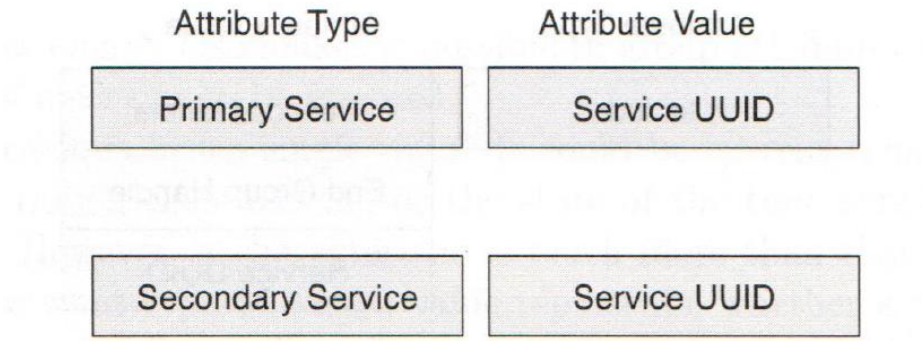



**Figure 10–21** Primary and secondary service declaration

# BLE: Services

- Including Services
  - Secondary services must be discovered separately
  - Each service can have zero or more Include attributes
  - Include attributes always immediately follow the service declaration and come before any other attributes for the service
  - The Include attribute definitions encompass the handle range for the referenced service along with the Service UUID
    - Enabling quick discovery of the referenced services, their group attributes, and type of services

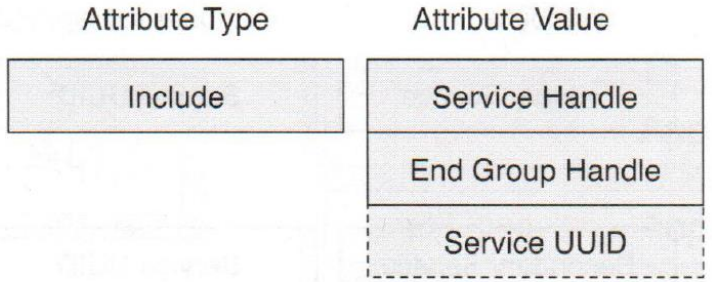



**Figure 10–22** The structure of the Include declaration

| Handle | Type | Value |
|---|---|---|
| 0x0001 | Primary Service | Service B |
| 0x0002 | Include | 0x0200, 0x0209, Service D |
| ... | ... | ... |
| 0x0100 | Primary Service | Service A |
| 0x0102 | Include | 0x0300, 0x0317, Service C |
| ... | ... | ... |
| 0x0200 | Secondary Service | Service D |
| ... | ... | ... |
| 0x0300 | Primary Service | Service C |
| ... | ... | ... |

**Figure 10–23** An example of an attribute database of Services A(C), B(D)

# BLE: Characteristics

- A characteristic is just a single value
- However, a characteristic needs to expose the following:
  - What type of data a value represents
  - Whether a value can be read or written
  - How to configure the value to be indicated or notified or broadcast
  - And what the value means
- To expose this additional information, a characteristic is composed of three basic elements:
  - Declaration: start of the characteristic and groups all the other attributes for the characteristic
  - Value: the actual value of the characteristic
  - Descriptors: the additional information or configuration of the characteristic

# BLE: Characteristics

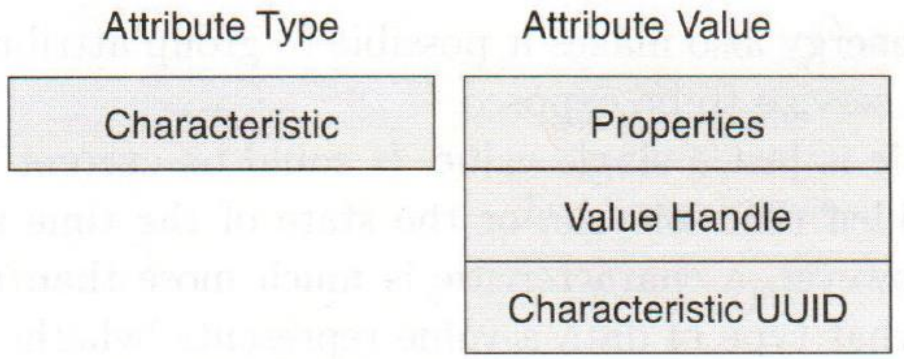



**Figure 10–24** Characteristic Declaration

- Characteristic Declaration starts a characteristic and contains three fields:
  - Characteristic properties: specifies if the characteristic value attribute can be read, written, notified, indicated, broadcasted, commanded, or authenticated in a signed write
  - Handle of the value attribute: the handle of the attribute that contains the value for the characteristic
  - Type of characteristic: The characteristic UUID is used to identify the type of characteristic value

# BLE: Characteristics

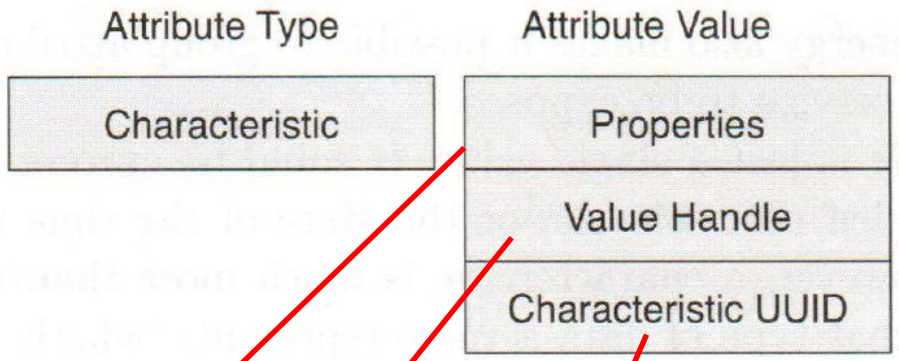



**Figure 10–24** Characteristic Declaration

| Handle | Type | Value |
|---|---|---|
| 0x0001 | Primary Service | GAP Service |
| 0x0002 | Characteristic | read write, 0x0003, Device Name |
| 0x0003 | Device Name | "Proximity Tag" |
| 0x0004 | Characteristic | read, 0x0005, Appearance |
| 0x0005 | Appearance | Tag |

**Figure 10–25** Characteristic example

# BLE: Characteristics

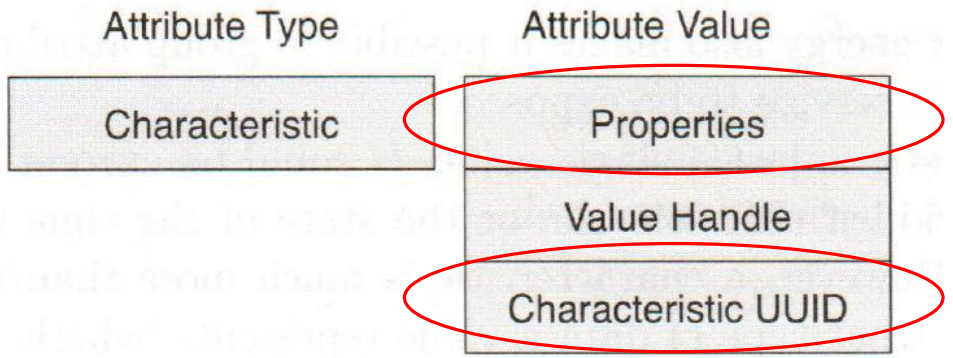
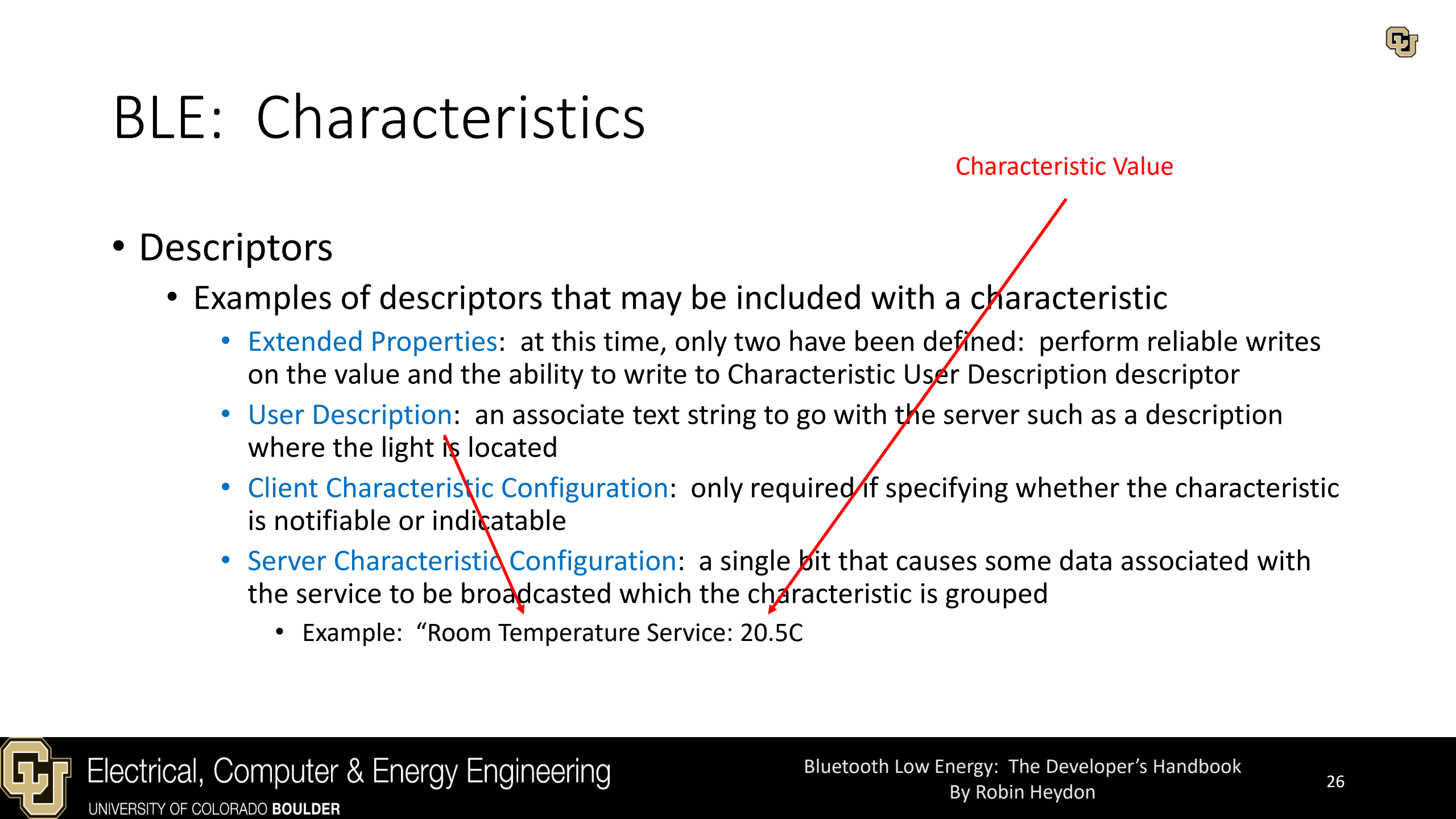



**Figure 10–24** Characteristic Declaration

- Characteristic Value
  - The characteristic value is an attribute with the type that must match the characteristic declaration's characteristic UUID field.
  - Types of actions that can be performed on the characteristic value is exposed by the characteristic declaration or in the characteristic extended properties descriptor
  - Characteristics have no behavior, so the service specification with which this characteristic is grouped should specify the behavior exposed by this instance of the characteristic

# BLE: Characteristics




- Descriptors
  - Examples of descriptors that may be included with a characteristic
    - Extended Properties: at this time, only two have been defined: perform reliable writes on the value and the ability to write to Characteristic User Description descriptor
    - User Description: an associate text string to go with the server such as a description where the light is located
    - Client Characteristic Configuration: only required if specifying whether the characteristic is notifiable or indicatable
    - Server Characteristic Configuration: a single bit that causes some data associated with the service to be broadcasted which the characteristic is grouped
      - Example: "Room Temperature Service: 20.5C

# BLE: Characteristics

Displayed value = characteristic value X $10^{exponent}$

- Descriptors
  - Examples of descriptors that may be included with a characteristic
    - Characteristic Presentation Format: a multiple-field value that contains the following information:
      - Format: similar to variable declarations in C such as int, unsigned_16, float, etc.
      - Exponent: base 10 exponent that is a signed integer
      - Unit: UUID defined in the assigned numbers document such as Temperature in Celsius
      - Namespace: single byte that determines which organization controls the descriptor field
      - Description: a 16-bit unsigned number that is like an adjective. For example, if the thermometer exposes two temperature characteristics, this field could specify whether that particular characteristic is “inside” or “outside”
    - Characteristic Aggregation Format: used to reference two characteristic values that are concatenated together into a “single” value
      - Example: GPS coordinates would have a complex characteristic single value that is actually composed of the latitude and longitude values

# BLE: Attribute Protocol

- A simple protocol by which an attribute client can find and access attributes on an attribute server
- The six structured basic operations are:
  - Request
  - Response
  - Command
  - Indication
  - Confirmation
  - Notification

# BLE: Attribute Protocol

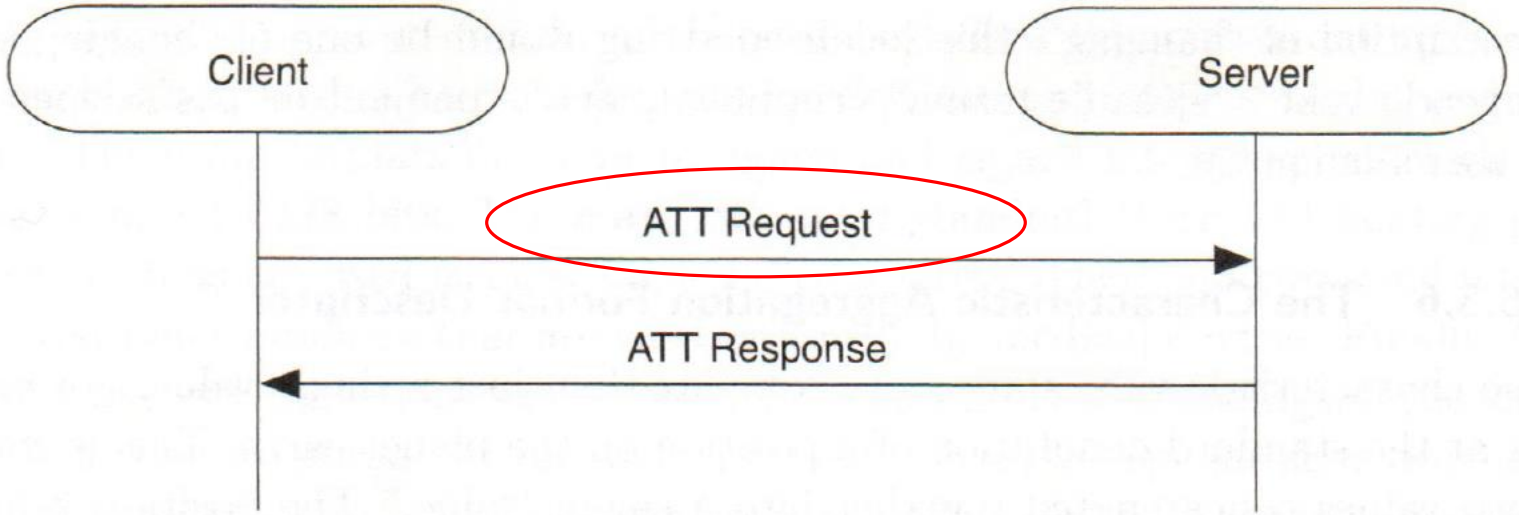



**Figure 10–26** An Attribute Protocol request

- The client can send only one request at a time
- The server only has two options for the response
  - A response directly associated with the request
  - An error message

# BLE: Attribute Protocol

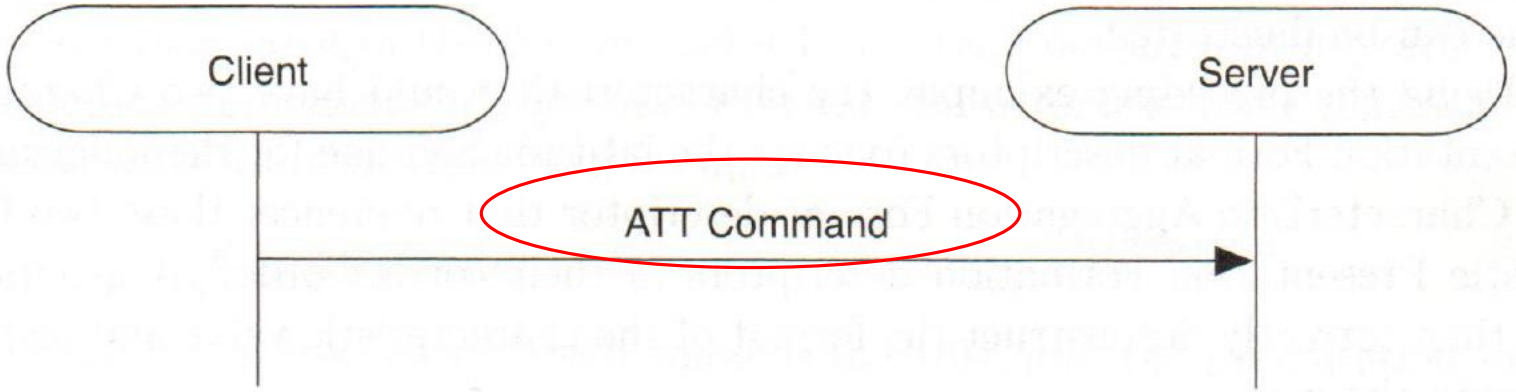



**Figure 10–27** An Attribute Protocol command

- The client sends a command to a server but receives no response
- The client sends a command when it wants the server to perform an action with no requirement for an immediate response

# BLE: Attribute Protocol

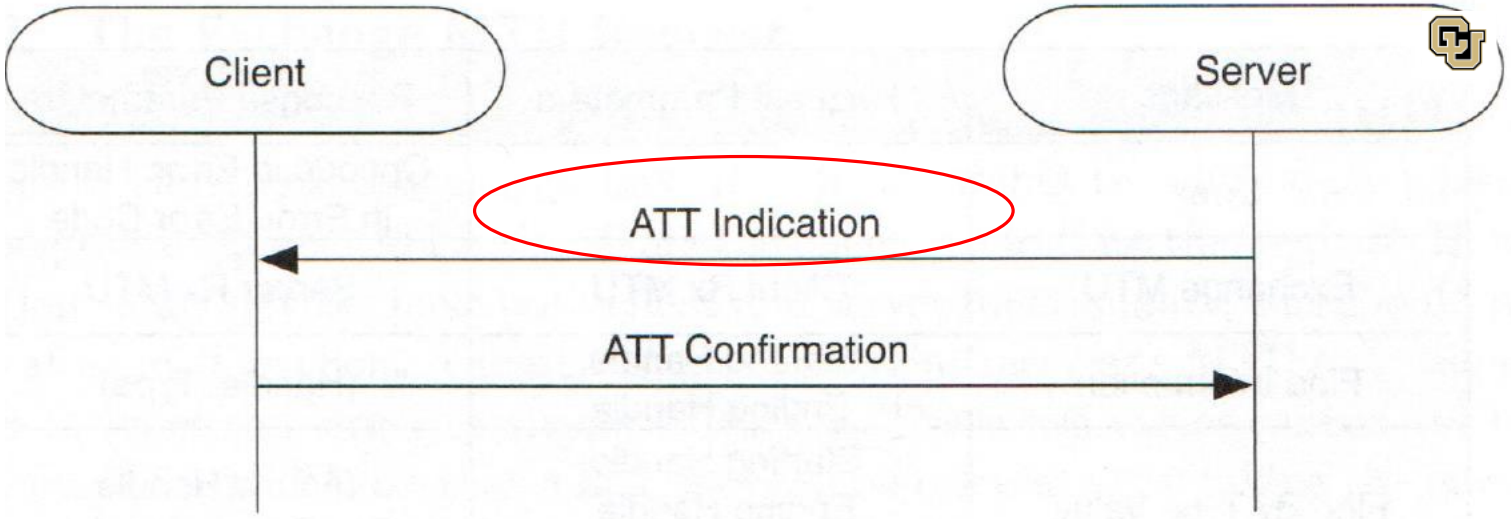



**Figure 10–28** An Attribute Protocol indication

- Indications are sent by a server to a client to inform the client that a particular attribute has given a value
- Indications are similar to requests in that only one indication can be sent and a confirmation is required by the client

# BLE: Attribute Protocol

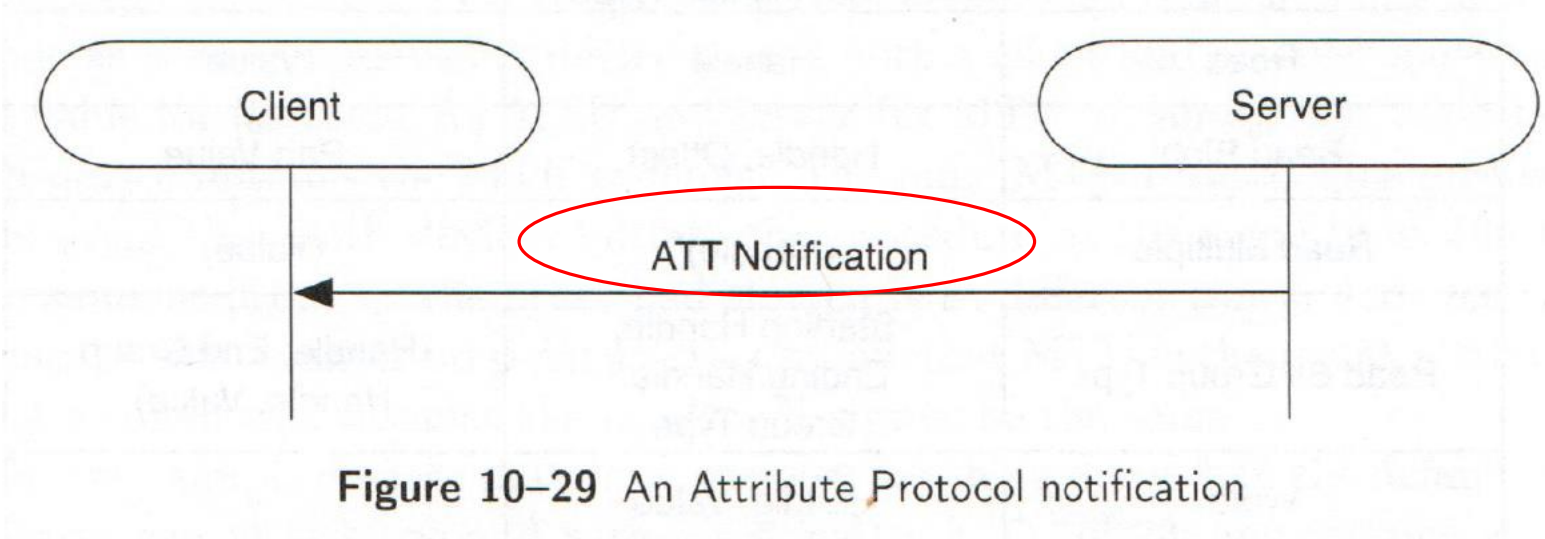



**Figure 10–29** An Attribute Protocol notification

- The server can send notification to a client to inform a client that a value of a particular attribute has changed, but does not require a confirmation

# BLE: Attribute Protocol

- Find Information Request and Response
  - Find handle and type information for a sequence of attributes
  - The requests includes two handles: a starting handle and an ending handle
  - Typically, the response packet is too short to complete this request, so the client must repeat the operation with incrementing the starting handle to be one more than the last response handle

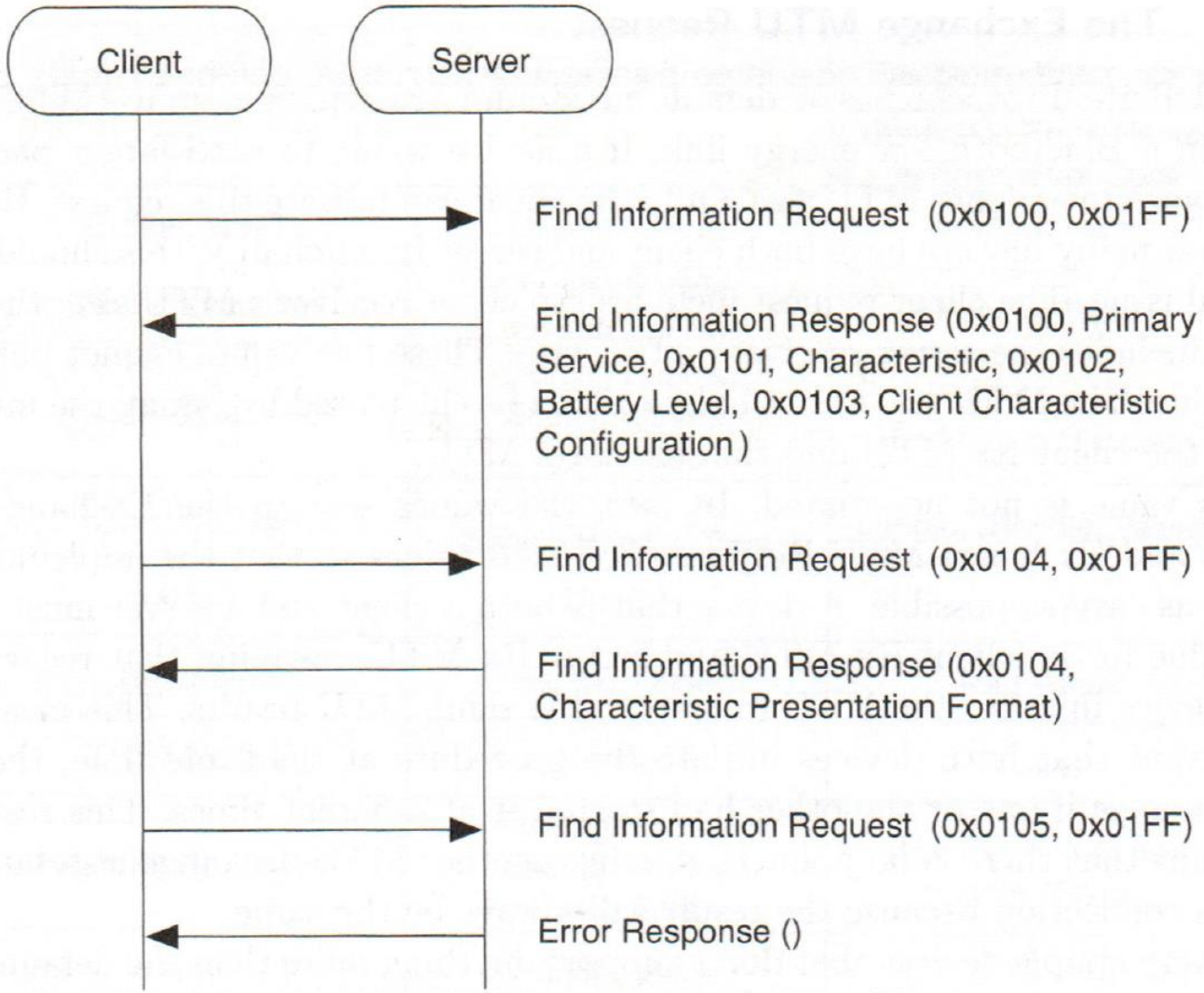



**Figure 10–31** The Find Information Request

# BLE: Attribute Protocol

- Find By Type Value Request
  - Finds all the attributes with a given type and value
  - The request includes a starting and an ending handle
  - Primary use of this request is to find all the primary services
  - Example:
    - A client can send a Find By Type Value Request with the type set to Primary Service and the value set to the UUID of the service
    - The response then includes the handle range of each instance of this primary service located
  - Secondary services are always included from other services, so the Ready By Type Request is used to discover these services

# BLE: Attribute Protocol

- Read By Type Request
  - Reads the value of an attribute within a range of handles
  - Used by the client who wants to know the attribute types within a range of handles, and not the handle
  - Each attribute within the handle range that has the requested type is returned
    - The response is a set of attribute handles and their associated values
  - Primarily used to search for included services as well as discovering all the characteristics of a service by using the Characteristic type
  - Example:
    - The client wants to quickly read the battery level of a device, read by type request set to “Battery Level” will respond with the attributes handle and values
    - No need to send a secondary read request using the attribute handle

# BLE: Attribute Protocol

- Read Request
  - Simplest Attribute Protocol request
  - Request includes a handle, and the response returns the value of the attribute identified by the handle
  - Only useful after the attribute's handle has been identified
- Read Blob Request
  - Blob comes from the database term meaning Binary Long Object
  - If the value of an attribute is longer than what can be contained within a Read Response, a Read Blob Request is used
  - The Read Blob Request includes the handle as well as a given offset into the attribute's value
  - Example:
    - The Read Blob Request can be used after a Read Request that returned the first 22 octets of the attribute value
    - The Read Blob Request would use the same handle with an offset of 22

# BLE: Attribute Protocol

- Read Multiple Request
  - Reads multiple attribute values in a single operation
  - The request includes a set of one more attribute handles
  - The response includes the value of these attribute handles in order they were requested
- Ready By Group Type Request
  - Takes a range of handles that the read will be considered over as well as an attribute type
  - The response will includes the handle of the read attribute and the last attribute for the grouping of attribute
  - Example:
    - If the grouping type is a Primary Service, the response will include the Primary Service attribute handle and the last handle of the last attribute of that grouping
    - This single request would return the entire range of handles for a Primary Service as well as the value of the Primary Service

# BLE: Attribute Protocol

- Write Request
  - The request includes a handle and the value written into that attribute
  - The response acknowledges that the value was written into the attribute
- Write Command
  - The command includes the handle of the attribute and the value to be written
  - There is no response
- Signed Write Command
  - Similar to the Write Command except it also includes an authentication signature

# BLE: Attribute Protocol

- Prepare Write Request and Execute Write Request
  - Used for two purposes
    - First, provide the ability to write long attribute values
    - Second, allow multiple values to be written in a single-executed atomic operation
  - The values prepared to be written are not written into the attribute until the Execute Write Request is received with the go-ahead to execute the writes
  - The Prepare Write Request includes the handle, offset, and part attribute value in a similar way as that of a Read Blob Request
  - This insures that all parts of the attribute are written at the same time
  - The Prepare Write Response includes the handle, offset, and part attribute value for the client to check and insure that the correct write was received

# BLE: Attribute Protocol

- Handle Value Notification
  - Used by the server to send a quick attribute state update to a client
  - The server provides the attribute handle and value
  - The serve can update the client with attribute values or notify the client of changes in a finite state machine
  - The client does not respond to these notifications
- Handle Value Indication
  - Similar to the Handle Value Notification in that it provides the attribute handle and value, but it requires a confirmation receipt from the client
  - Because of the confirmation, indications are considered reliable while notification are not

# BLE: Attribute Protocol

- If the Client wants to read multiple attribute values, can it send these multiple read requests back to back?
  - Why not?
- How would the Client read multiple attribute values?

# BLE: Attribute Protocol

- Error Response
  - An Error Response can be sent by a device whenever a request cannot be achieved
  - The Error Response includes all the information about the request that cause the error, the attribute on which the request failed, and why the error was generated
  - Whenever the client receives an error response, it must assume that the error response is for the last request that was sent
    - Thus, and error response is another way to close the request's transaction

# BLE: Attribute Protocol

- Possible Error Responses are:
  - Invalid Handle
  - Read Not Permitted
  - Write Not Permitted
  - Invalid PDU
  - Insufficient Authentication
  - Request Not Supported
  - Invalid Offset
  - Insufficient Authorization
  - Prepare Queue Full
  - Attribute Not Found
  - Attribute Not Long
  - Insufficient Encryption Key Size
  - Invalid Attribute Value Length
  - Unlikely Error
  - Insufficient Encryption
  - Unsupported Group Type
  - Insufficient Resources
  - Application Errors

# BLE & Security

- Definitions
  - Authentication:  Prove that the device is who / what it claims to be
    - Two basic methods:
      - Initial authentication and sharing of a secret
      - Re-authorization using a previously shared secret
    - In BLE, authentication is performed in three different ways
      - At initial pairing, an authentication algorithm is used to authenticate the devices
        - This allows shared secrets to be stored, and the devices are said to be “bonded”

# BLE & Security

- Definitions
  - In BLE, authentication is performed in three different ways (continued)
    - Upon reconnecting to a device in which the devices have previously bonded, one of the devices sends a signed command to the other device to authenticate that it knows the shared secret
    - Since the signature is created using the shared secret exchanged during bonding, it cannot be falsified
    - The signed command process must include some revolving algorithm to prevent replay attacks.
  - When reconnecting to a device that has previously been bonded, either device can initiate encryption
    - From the moment onwards, all packets will incorporate Message Integrity Check (MIC) value that authenticate the sender of the message

# BLE & Security

- Definitions
  - Authorization:  Assigning of permission to something
    - Documentation that provides authorization
    - Authorization that is actioned directly
  - Authentication ≠ Authorization

  - Integrity:  Internal consistency and lack of data corruption
    - Cyclic Redundancy Checks (CRC) are used to protect against bit changes, but they are typically too week to be considered a security measure
      - Valid CRD ≠ Integrity

# BLE & Security

- Confidentiality:  the intent to keep something secret
  - In BLE, confidentiality refers to that even if a third-party receives a message, they cannot decode it
  - In BLE, confidentiality is provide via encryption

- Privacy:  The ability to prevent others from recognizing you by the device you are carrying and not allowing them to track your movement throughout space

# BLE & Security

- A key in BLE is shorthand for shared secret
- There are five types of keys in BLE
  - Temporary Key: A temporary key used during the pairing process that is determined by the pairing algorithm
    - "Just Works" is a mode designed to enable pairing of devices with limited user interface
      - No authentication is being performed
      - It is open to the man-in-the-middle attack
    - "Passkey Entry" is a mode used when the user interface on both devices allow the display or entry of a number value, 0 – 999999
      - With only 1 in 1,000,000 of a chance of a man-in-the-middle attack, this method can generate an authenticated key

# BLE & Security

- Temporary Key: A temporary key used during the pairing process that is determined by the pairing algorithm (continued)
  - "Out of Band" is a mode both devices have information that has been acquired by using another technology than Bluetooth

- Short-Term Key: Is a key used for encrypting a connection the very first time two devices pair
  - Utilizes three pieces of information to generate the STK:
    - The Temporary Key
    - Srand
    - Mrand

# BLE & Security

- Long-Term Key: The key distributed once the initial pairing procedure has encrypted the connection
  - Upon reconnection to a previously paired and bonded device, the LTK is used to encrypt the link
- Identity Resolving Key: Give a device that knows a peer device the ability to resolve (work out) a peer device's identity
  - IRK enables a device to randomly change its address and become "private"
  - IRK and Authentication are typically combined to insure that the correct device is at the "random address"

# BLE & Security

- Connection Signature Resolving Key (CSRK): Provides the receiving device the ability to resolve a signature and therefore authenticate the sender of a message
  - The CSRK is distributed over an encrypted link
  - But, the signed messages do not

# BLE & Security

- Pairing involves the following:
  - Exchange of pairing information
  - Authentication of the link
  - Key distribution (becoming “bonded”)

  - Exchange of information
    - First, the device determines the input and output capabilities
      - Depending on the I/O capabilities, the Temporary Key mode will be defined
      - The first secured message is the pairing request that will be returned

# BLE & Security

- After resolving the Temporary Key mode
  - The first secured message is the pairing request that will be returned
  - This message will include:
    - What the authentication requirements are, if any, such as whether bonding is enabled and whether man-in-the-middle protection is required
    - List of keys that are being requested
    - The Temporary Key can be generated
  - Once the Temporary Key is generated, the Short-Term Key can be generated to enable the passing of the Long-Term Key, Integrity Resolving Key, and the Connection Signature Resolving Key