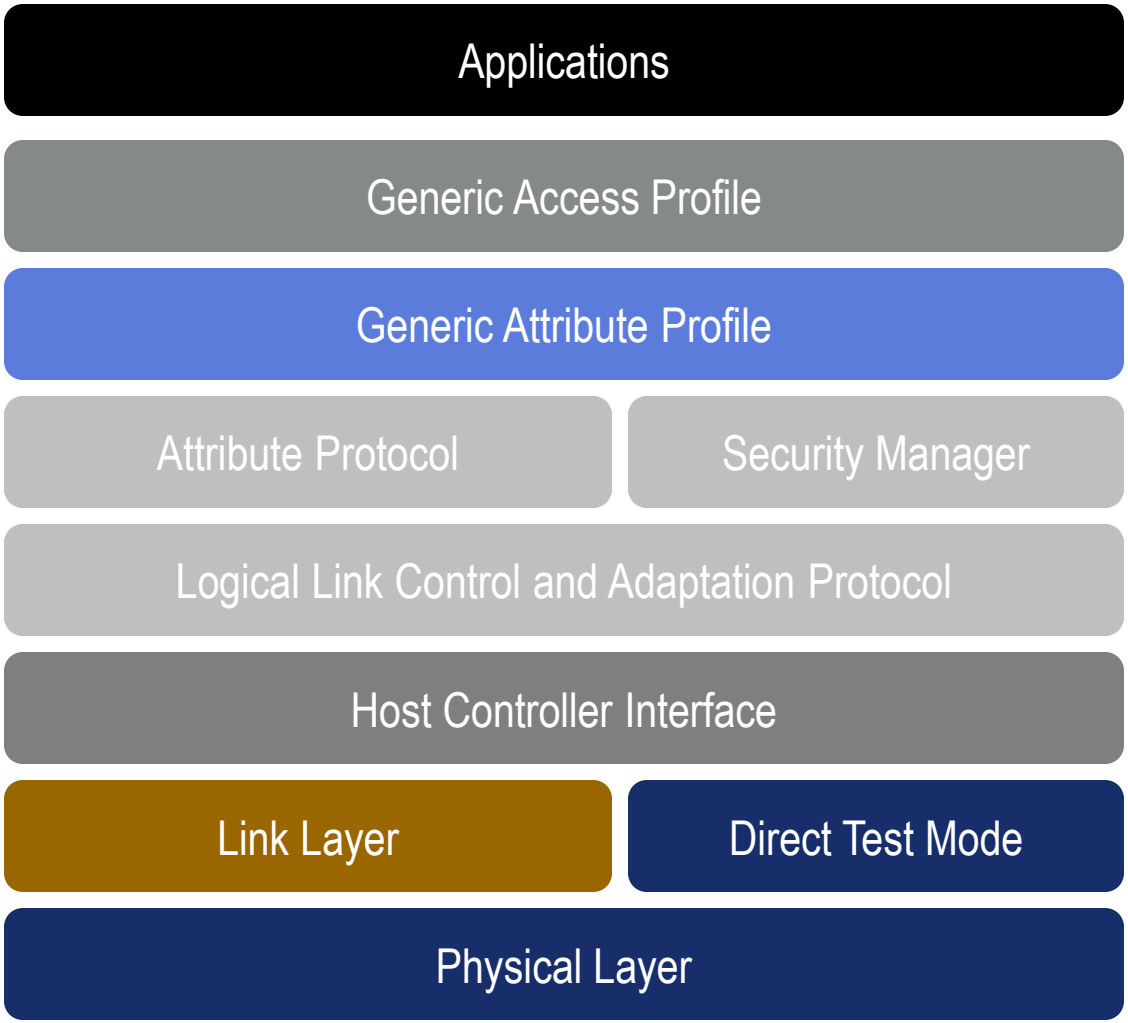
ATTRIBUTE PROFILE (ATT) AND GENERIC ATTRIBUTE PROFILE (GATT)

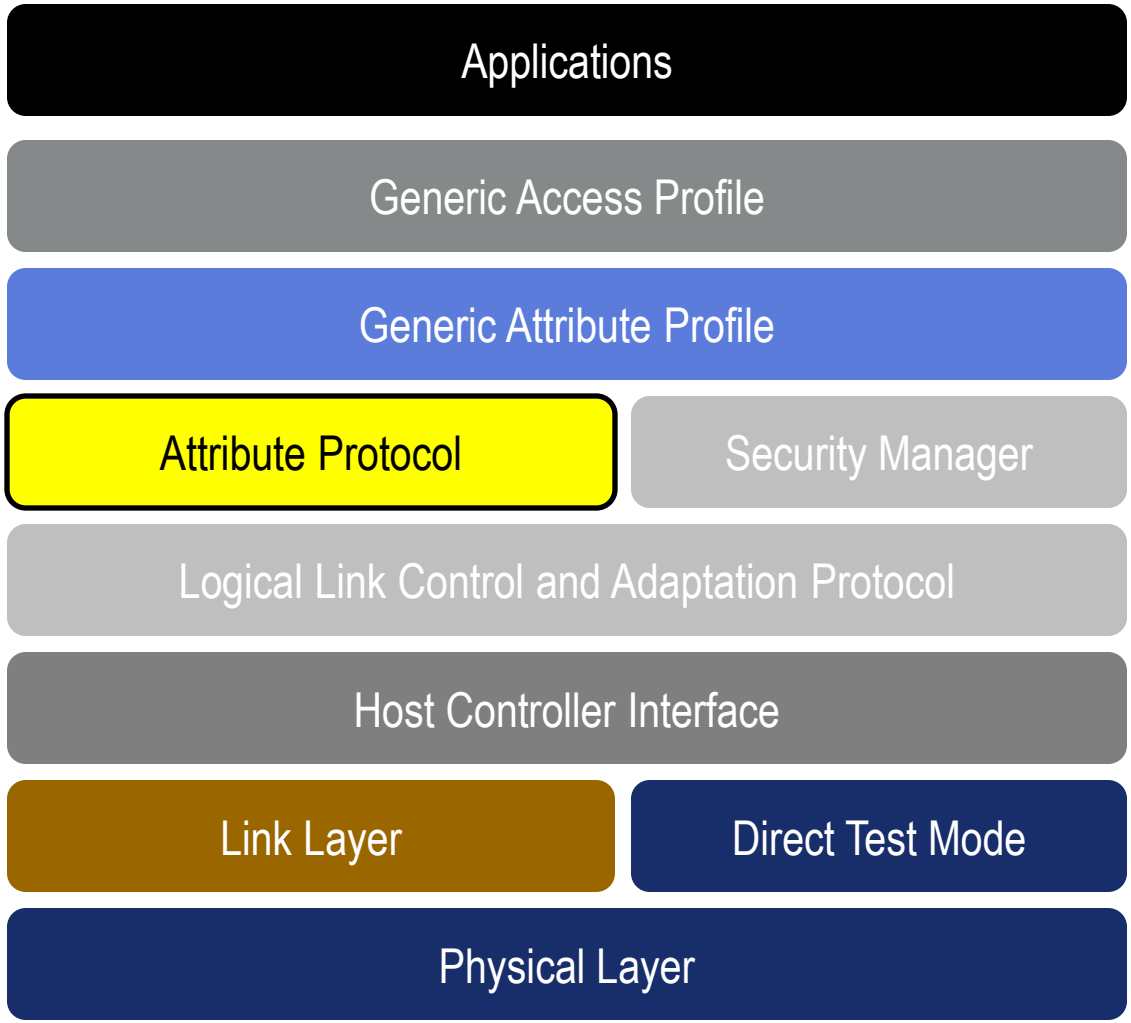
ATT & GATT Overview

- Supported on both LE and BR/EDR
- LE Transport
  - ATT/GATT mandatory for LE
  - ATT/GATT primary mechanism for service discovery
  - Fixed channel on L2CAP
  - ATT packets optimized for smaller packets used in LE
- BR/EDR Transport
  - ATT/GATT optional on BR/EDR
  - SDP primary mechanism for service discovery
  - Uses connection oriented L2CAP channel

Architectural Block Diagram



ATTRIBUTE PROTOCOL (ATT)

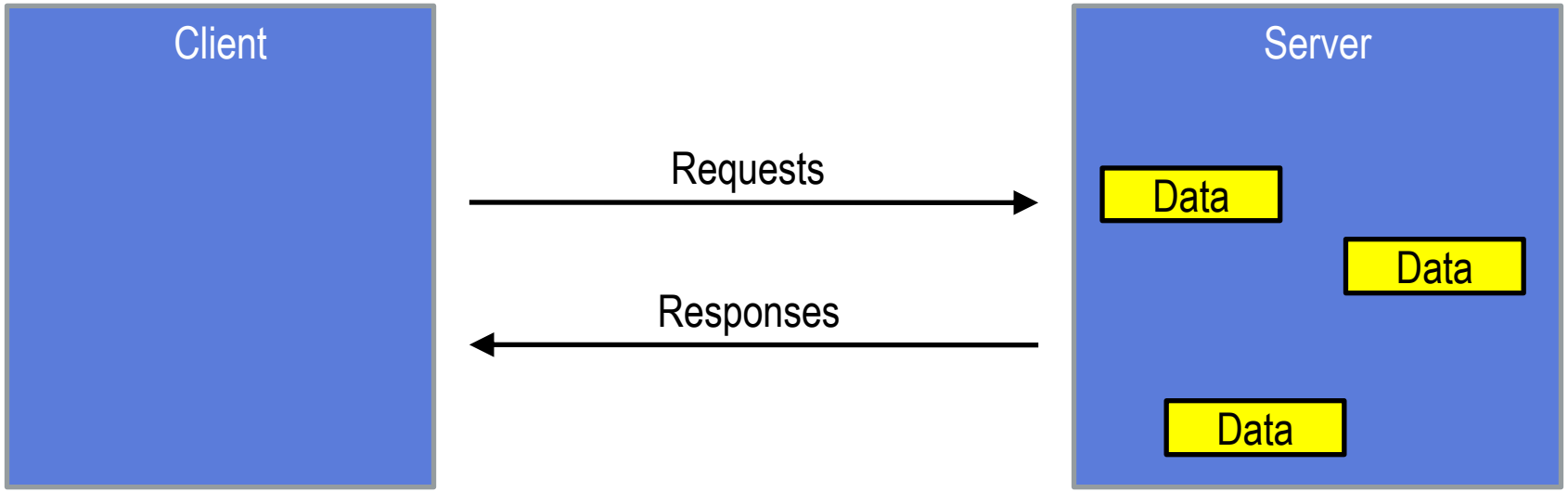


ATTRIBUTE PROTOCOL (ATT)

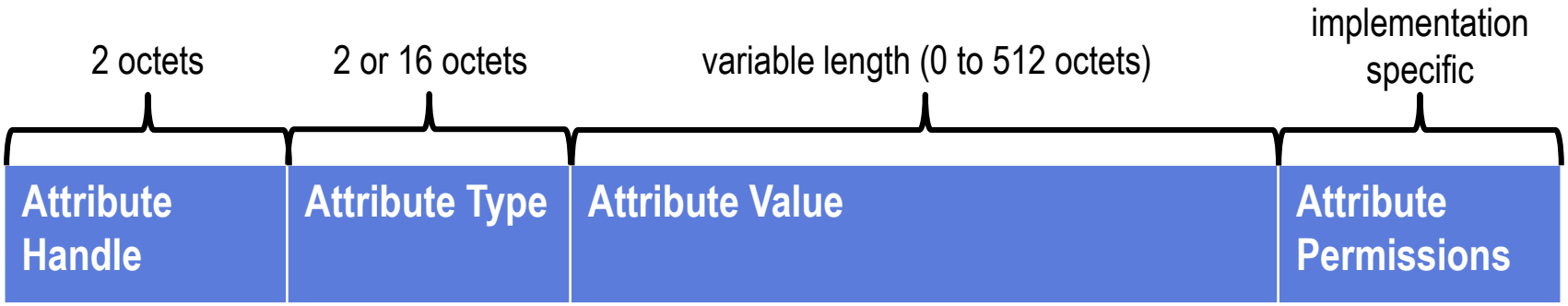
- A protocol for discovering, reading, and writing attributes on a peer device.
- Implements the peer-to-peer protocol between an attribute server and an attribute client.
- Client-Server Architecture
  - Servers have data
  - Clients request data to/from servers
  - Clients communicate with an ATT server on a remote device over a dedicated fixed L2CAP channel
- Protocol Methods
  - Client: Request, command, confirmation
  - Server: Response, notification, indication

CLIENT-SERVER ARCHITECTURE

- Servers have data, Clients want to read/write this data
- Servers use Attributes as container to send/receive data



LOGICAL ATTRIBUTE REPRESENTATION



- Attributes are composed of a handle, type, value and permissions
- An Attribute value is an array of octets, 0 to 512 octets in length can be fixed or variable length

ATTRIBUTES Handles

- Each attribute has a “handle”
  - Used to address an individual attribute by a client

- Clients use handles to address attributes
  - Read (0x0022) => 0x04 ; Read (0x0098) => 0x0802

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

ATTRIBUTE TYPE

- Attributes type is a «UUID»
  - Attribute type is an assigned number (16 bit) or implementation specific (128 bit)
    - > Attribute UUIDs follow same rules as those in SDP
  - Attribute type defines how the value is used
  - Attribute type is defined in the GATT, GAP or GATT based profile, service or characteristic specifications.

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

ATTRIBUTE PERMISSIONS

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

ATTRIBUTE PERMISSIONS

- Attribute Handles are public information
- Attribute Types are public information
- Attribute Values can be protected
- A server may not reveal any values that are protected to a client that it does not “trust” enough
- Server responds with a reason/notification (Message)– not with a value
  - Insufficient Authentication / Authorization / Key Size
  - Read / Write Not Permitted

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 transaction 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

ATOMIC OPERATIONS

- Each request / command is an “atomic operation”*
  - Cannot be affected by another client at the same time

- If link is disconnected halfway through a transaction
  - Value of attribute(s) undefined
  - There is no “rollback” or “transactional processing”

* An operation during which a processor can simultaneously read a location and write it in the same bus operation. This prevents any other processor or I/O device from writing or reading memory until the operation is complete.
Atomic implies indivisibility and irreducibility, so an atomic operation must be performed entirely or not performed at all.

MTU SIZES

- Over LE
  - Attribute protocol uses a fixed channel
  - MTU “exchanged” by ATT
    > ATT_MTU = min (Server_Rx_MTU, Client_Rx_MTU)
  - ATT_MTU is symmetrical (same on client / server)

- Over BR/EDR
  - Attribute protocol uses a dynamic channel (fixed PSM - Protocol Service Multiplexer)
  - MTU negotiated by L2CAP

ATTRIBUTE PDU FORMAT

- Attribute Opcode
  - Bit 6-0 : Method
  - Bit 7 : Authentication Signature Flag = 0

- Can append Signed Data to some methods

ATTRIBUTE PDU FORMAT

- Attribute Opcode
  - Bit 6-0 : Method
  - Bit 7 : Authentication Signature Flag = 1

- Can append Signed Data to some methods

ATTRIBUTES & DESCRIPTIONS

Name	Description
Error Response	Something was wrong with a request
Exchange MTU Request / Response	Exchange new ATT_MTU
Find Information Request / Response	Find information about attributes
Find By Type Value Request / Response	Find specific attributes
Read By Group Type Request / Resposne	Find specific group attributes and ranges
Read By Type Request / Response	Read attribute values of a given type
Read Read / Response	Read an attribute value
Read Blob Request / Response	Read part of a long attribute value
Read Multiple Request / Response	Read multiple attribute values
Write Command	Write this – no response
Write Request / Response	Write an attribute value
Prepare Write Request / Response	Prepare to write a value (long)
Execute Write Request / Response	Execute these prepared values
Handle Value Notification	Notify attribute value – no confirmation
Handle Value Indication / Confirmation	This attribute now has this value

MTU EXCHANGE

- **`Exchange MTU (Client Rx MTU) => (Server Rx MTU)`**

- Only used on LE (not on BR/EDR)
  - ATT_MTU = min (Client Rx MTU, Server Rx MTU)

- Can only be sent once during a connection
  - Only be initiated by client
  - Optional to initiate this – if you are happy with 23, don't
  - Server has no control over if this is done

FINDING INFORMATION

- **`Find Information (Starting Handle, Ending Handle) => (format, [Handle, Type]*)`**

- Handles / Types are public information
  - Find Information used to find this information

- Responds with a list of attribute handles and their types can only send all 16 bit or all 128 bit UUIDs in response

- If mix of 16/128 UUIDs, multiple requests must be used each with different format

FINDING INFORMATION

- **`Find By Type Value (Starting Handle, Ending Handle, Attribute Type, Attribute Value) => ([HandlePair]*)`**

- Returns
  - The handle of all found attributes with type and value matching exactly

  AND
  - The handle of the last attribute in the attribute group

READING ATTRIBUTES

- **`Read By Group Type (Starting Handle, Ending Handle, UUID) => (Length, [Handle:EndGroupHandle:Value]*)`**

- Reads the value of each attribute of a given type in a range
  - Responds with handle:end group handle:value tuples
  - Allows reading group semantics

- If multiple values have the size length, then many can be returned in a single response

READING ATTRIBUTES

- **`Read By Type (Starting Handle, Ending Handle, UUID) => (Length, [Handle:Value]*)`**

- Reads the value of each attribute of a given type in a range
  - Responds with handle:value pairs

- If multiple values have the size length, then many can be returned in a single response

READING ATTRIBUTES

- **`Read (Handle) => (Value)`**
- Simple?
- Only works up to ATT_MTU – 1 octets

READING LONG ATTRIBUTES

- **`Read Blob (Handle, Offset) => (Part Value)`**

- Can be used to read very long attributes

- If ATT_MTU = 23 (default on LE)
  - A 512 octet value would require 24 transactions to read

- If ATT_MTU = 48 (default on BR/EDR)
  - A 512 octet value would require 11 transactions to read

READ MULTIPLE

- **`Read Multiple ([Handle]*) => ([Value]*)`**
- Used to read multiple values at the same time
- Can only be used if len (value) < ATT_MTU – 1
- Can only be used if size of each value is known except last value

WRITING ATTRIBUTES

- **`Write Command (handle, value) =>`**

- I want this attribute set to this value NOW with no response

- Used for “sending commands” to a “state machine”

- Can also be signed if need authentication with cost of encryption setup / confidentiality requirements

WRITING ATTRIBUTES

- `Write (handle, value) => ()`

- Simple?

- Includes a Response to allow:
  - Flow control of transactions – no buffer overflows
  - Confirm that the value has been written

COMPLEX WRITES

- **`Prepare Write (handle, offset, value) => (handle, offset, value)`**
- **`Execute Write (exec/cancel) => ()`**

- Prepared writes are queued in server until they are executed
  - Queue size can be discovered
  - Response includes value again (double checking)
  - Offset parameter allows writes to long attributes

- Execution is a single atomic transaction

SERVER INITIATED METHODS

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

- Server notifies client of an attribute with a new value

- Can be sent at any time, no flow control
  - Inherently unreliable – can be dropped by buffer overflow

- Typically used to notify client of state updates
  - Turned on / off by clients using Generic Attribute Profile

SERVER INITIATED METHODS

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

- Server indicates to client an attribute with a new value
  - Client must confirm receipt before server can send again

- Used when flow control is important
  - Or when confirmation of state change is required

ERROR RESPONSE

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

- Any request can cause an error
  - Response must always be sent
  - Error response includes information about error

- Opcode from the request – what request caused this error

- Handle from the request – what handle caused this error

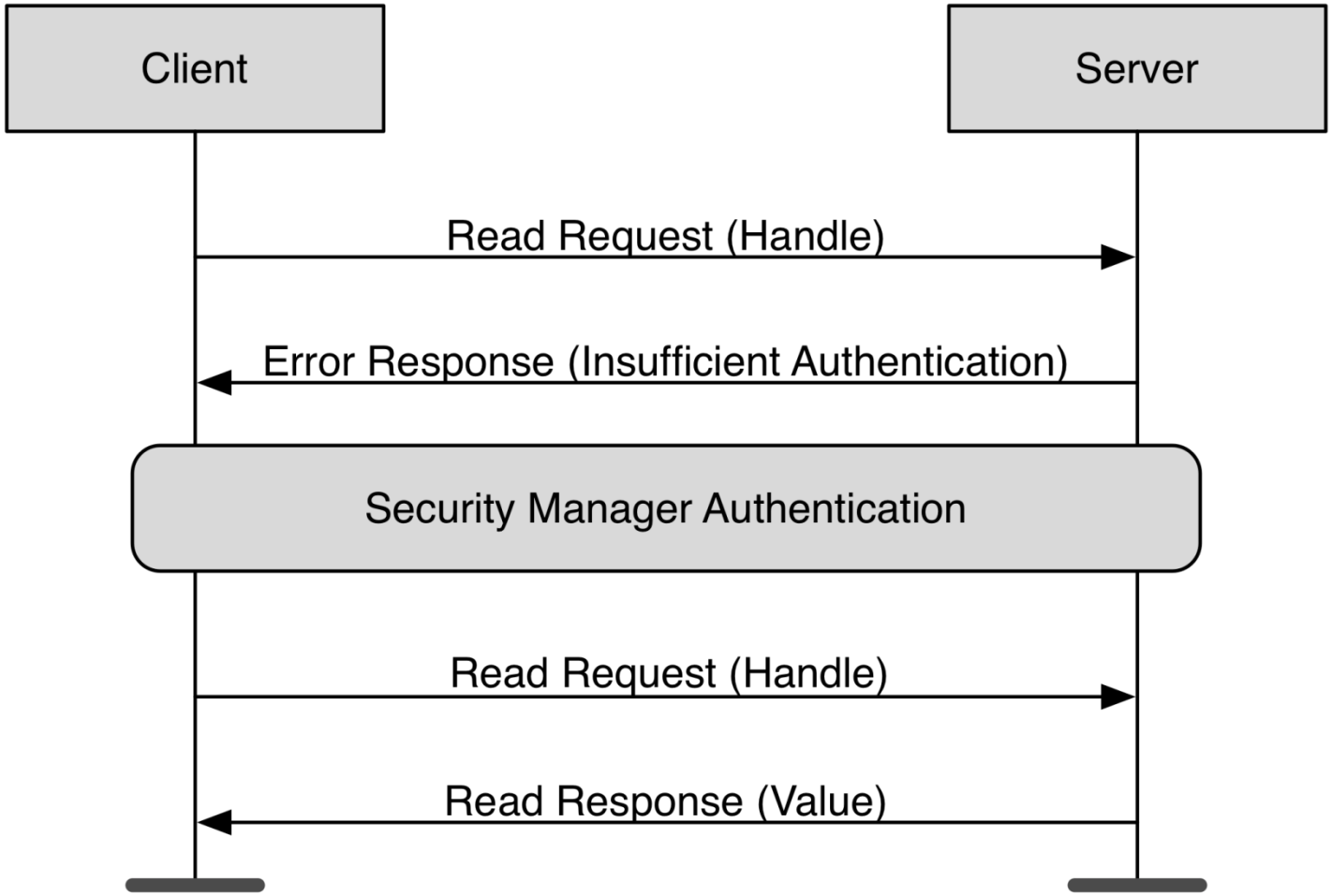
- Error Code – reason why this error is raised

ERROR TYPES/DESCRIPTIONS

Name	Description
Invalid Handle	for example handle = 0x0000
Read Not Permitted	not readable attribute : permissions
Write Not Permitted	not writeable attribute : permissions
Invalid PDU	PDU was invalid – wrong size?
Insufficient Authentication	needs authentication : permissions
Request Not Supported	server doesn't support request
Invalid Offset	offset beyond end of attribute
Insufficient Authorization	need authorization : permissions

Name	Description
Prepare Queue Full	server has run out of prepare queue space
Attribute Not Found	no attributes in attribute range found
Attribute Not Long	should use Read requests
Insufficient Encryption Key Size	needs encryption key size : permissions
Invalid Attribute Value Length	value written was invalid size
Unlikely Error	something went wrong – oops
Insufficient Encryption	needs encryption : permissions
Application Error	application didn't like what you requested

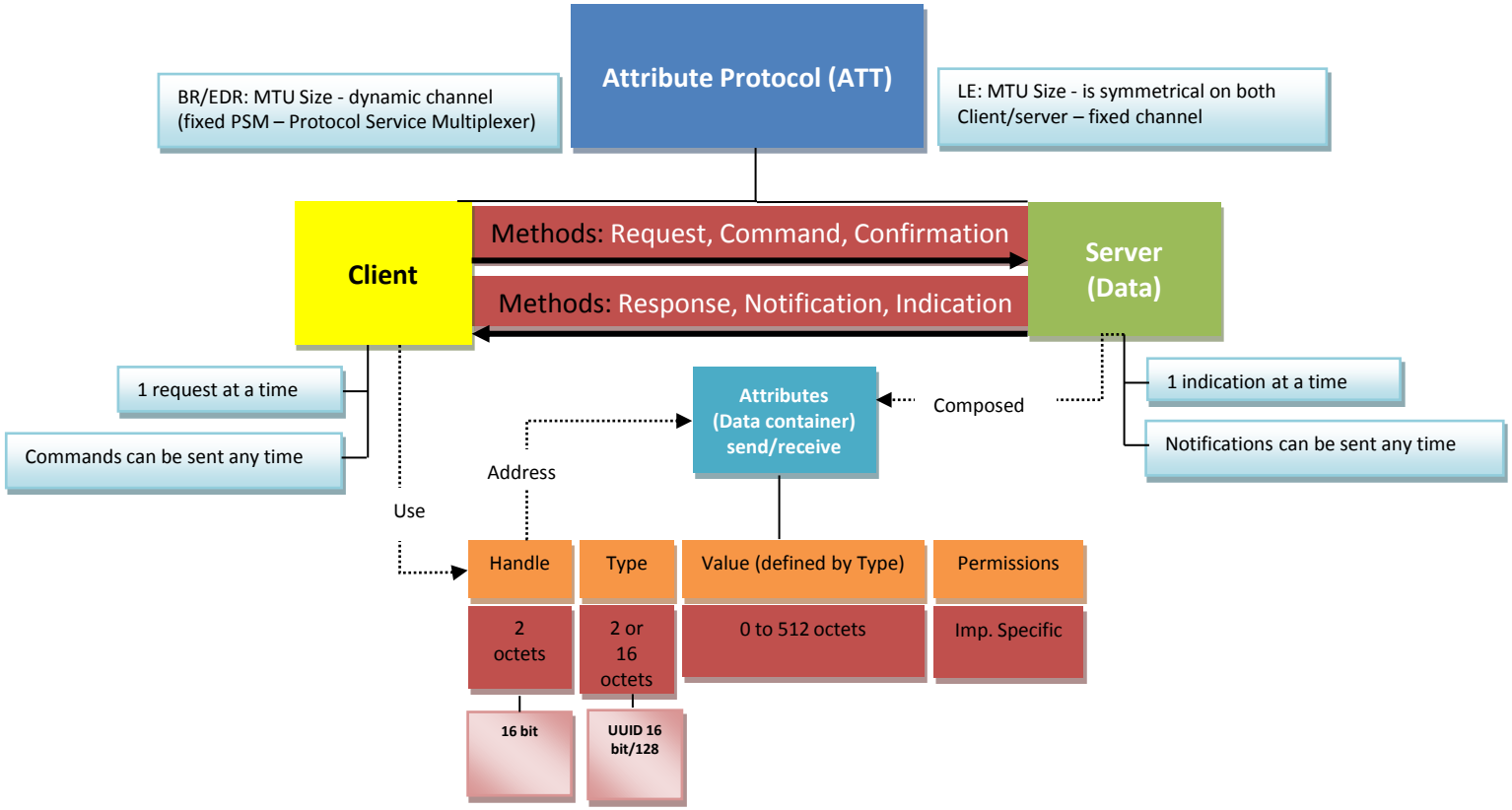
SECURITY CONSIDERATIONS



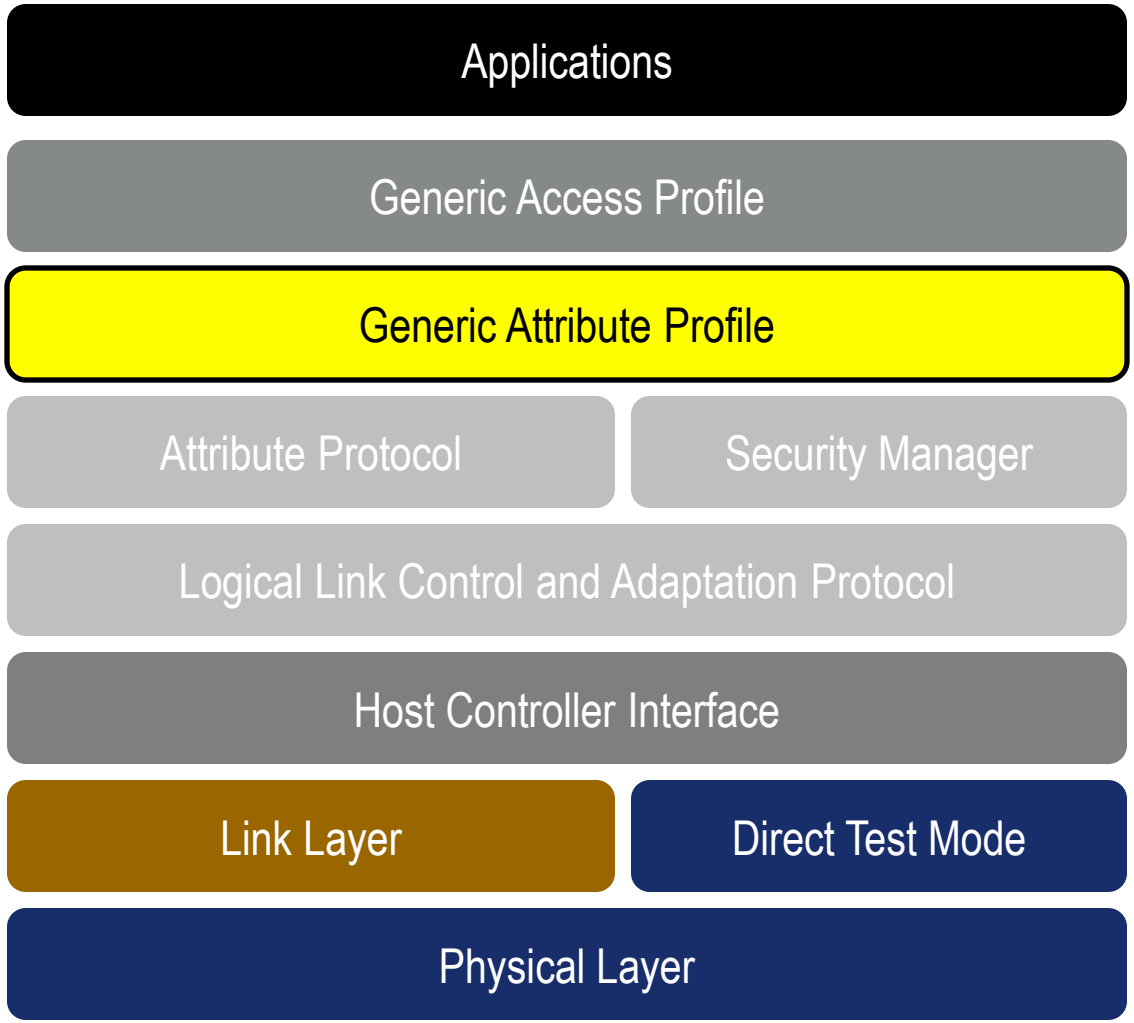
ATTRIBUTE PROTOCOL SUMMARY

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

ATTRIBUTE PROTOCOL SUMMARY



Generic Attribute Profile (GATT)



GENERIC ATTRIBUTE PROFILE (GATT)

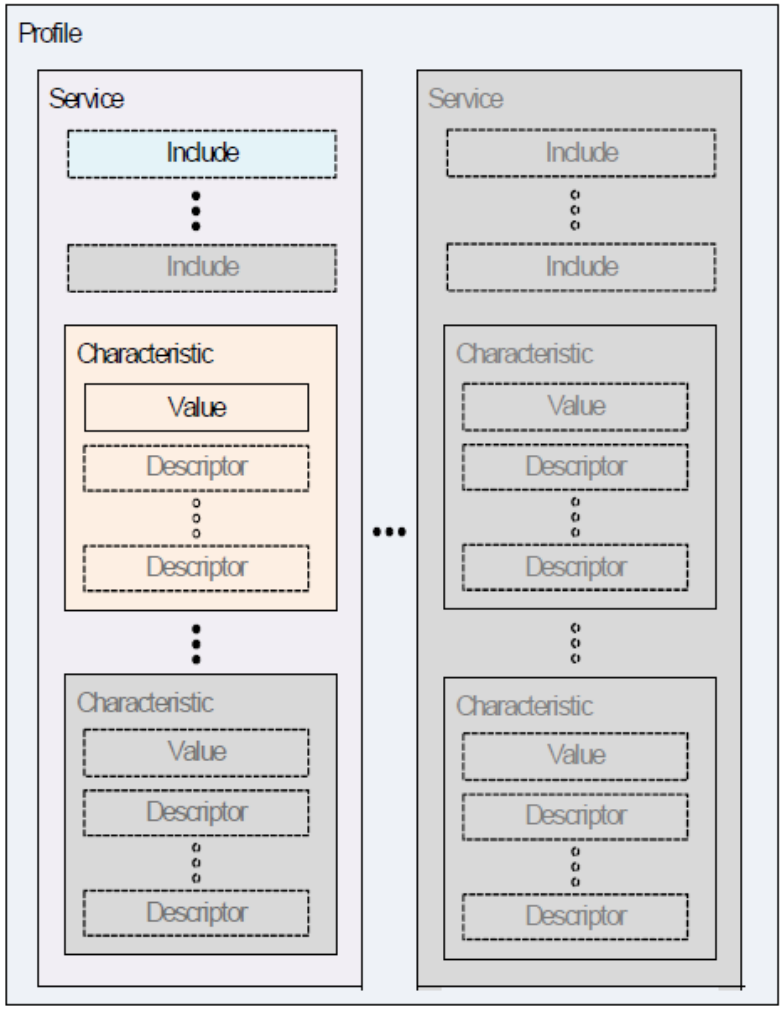
- Built on top of the ATT.
- Establishes common operations and a framework for the data transported and stored by ATT.
- The Generic Attribute Profile (GATT) block represents the functionality of the attribute server and, optionally, the attribute client.
- Describes the hierarchy of services, characteristics and attributes used in the attribute server.
- Provides interfaces for discovering, reading, writing and indicating of service characteristics and attributes.

GENERIC ATTRIBUTE PROFILE (GATT)

- GATT & ATT are not transport specific and can be used in both BR/EDR and LE.
- GATT & ATT are mandatory to implement in LE since they are used for *LE profile service discovery*
- Defines concepts of:
  - Service Group
  - Characteristic Group
  - Declarations
  - Descriptors
- Does not define rules for their use
  - This is separate but essential to understand

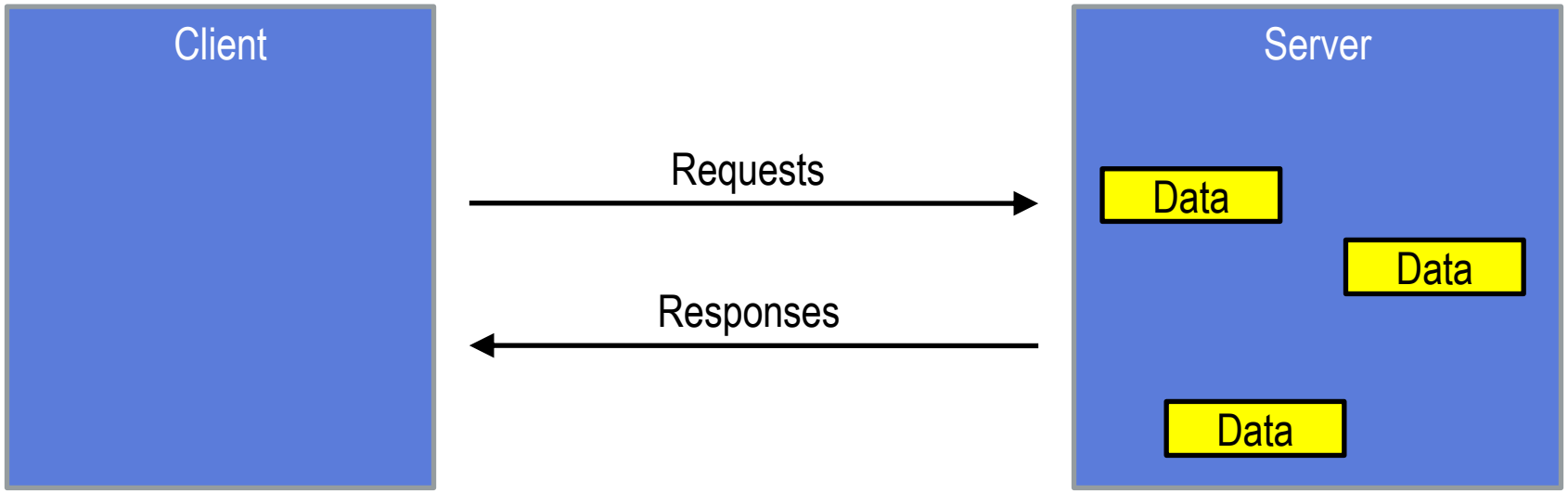
GENERIC ATTRIBUTE PROFILE (GATT) Hierarchy

- 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.
- 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 , 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.



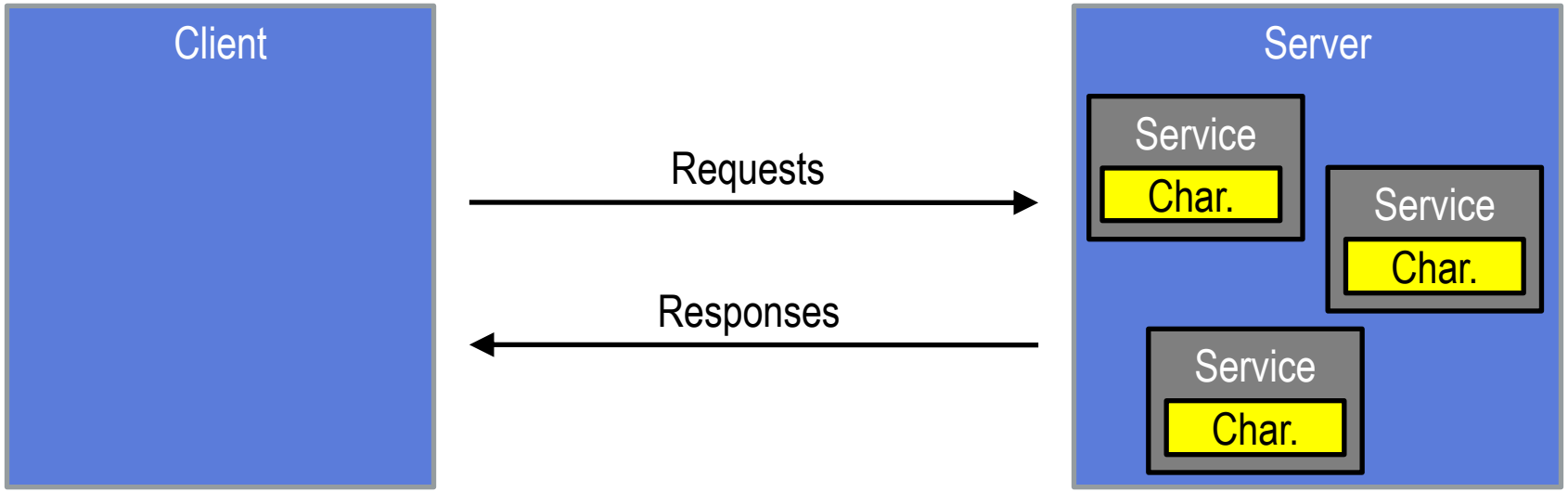
CLIENT-SERVER ARCHITECTURE

- Same client-server architecture as Attribute Protocol

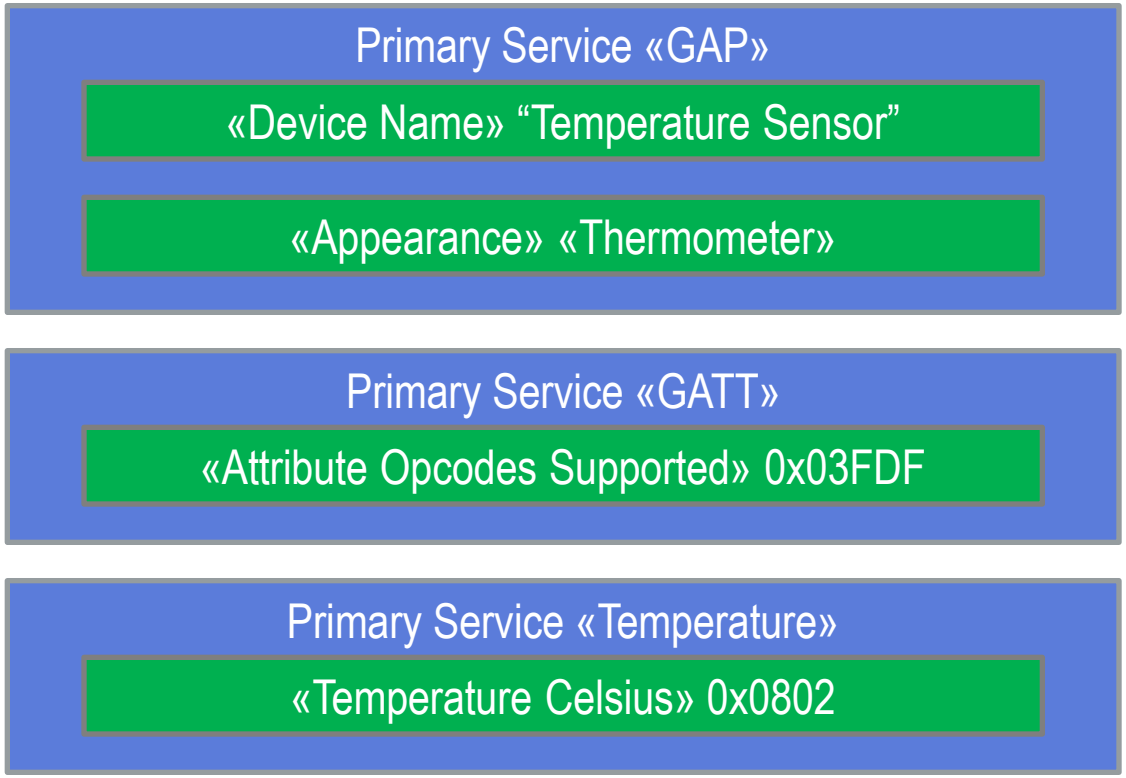


CLIENT SERVER ARCHITECTURE

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



GENERIC ATTRIBUTE PROFILE IMPOSES STRUCTURE



WHAT IS A SERVICE?

- A service is:
  - Defined in a “Service Specification”
  - Collection of characteristics
  - References to other services

- Service Declaration
  - Includes Characteristics

Two types of service:

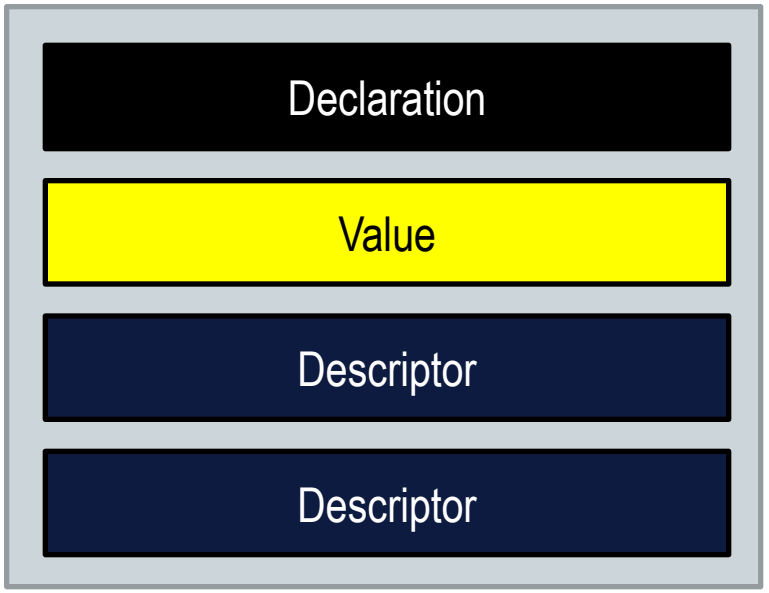
- Primary Service
  - A primary service is a service that exposes primary usable functionality of this device. A primary service can be included by another service.

- Secondary Service
  - A secondary service is a service that is subservient to another secondary service or primary service. A secondary service is only relevant in the context of another service.

WHAT IS A CHARACTERISTIC?

- It's a value, with a known type, and a known format defined in a "Characteristic Specification"

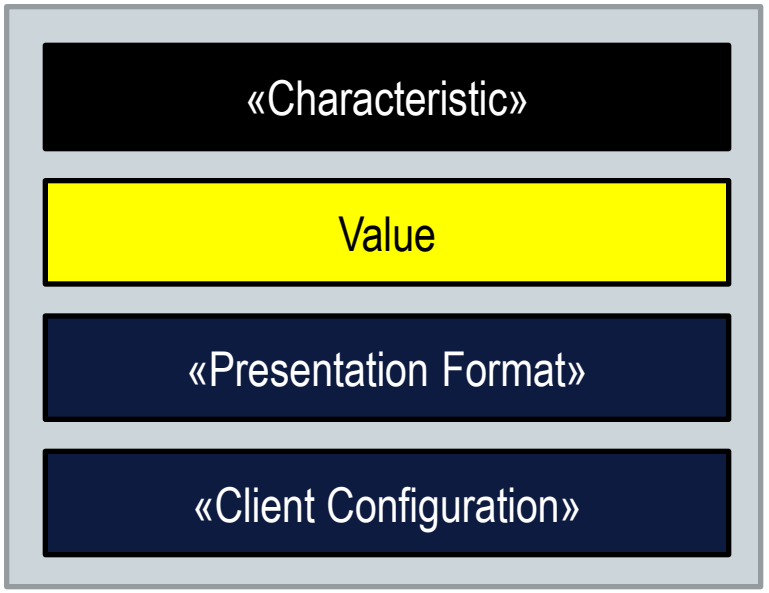
- Characteristic Declaration
  - Characteristic Value
  - Characteristic Descriptors



WHAT IS A CHARACTERISTIC?

- Characteristics are grouped by «Characteristic»
  - Value attribute is always immediately after «Characteristic» followed by descriptors

- Descriptors
  - Additional information
  - Any number
  - Any order
  - Can be vendor specific



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*

«Include» DECLARATION

- Allows services to reference other services

- Value is reference to another Service:
  - Service Handle
  - End Group Handle
  - Service UUID

Type	Value	Permissions
«Include»	Service Handle End Group Handle Service UUID	Read only, no authentication, no authorization

«Primary Service» DECLARATION

- Groups attributes for a primary service
  - Followed by «Include»
  - Then followed by «Characteristic»

- Attribute value is UUID for Service
  - e.g. «GAP», «GATT», «Temperature», «Battery», etc…

Type	Value	Permissions
«Primary Service»	UUID for Service	Read only, no authentication, no authorization

«Secondary Service» DECLARATION

- Groups attributes for a secondary service
  - Followed by «Include»
  - Then followed by «Characteristic»

- Attribute value is UUID for Service
  - e.g. «GAP», «GATT», «Temperature», «Battery», etc…

Type	Value	Permissions
«Secondary Service»	UUID for Service	Read only, no authentication, no authorization

«Characteristics» DECLARATION

- Groups attributes for a characteristic within a service
  - Followed by Characteristic Value attribute, descriptors

- Attribute Value is:
  - Properties for characteristic value (b,r,c,w,n,i,a,e)
  - Handle of characteristic value
  - Type of characteristic

Type	Value	Permissions
«Characteristic»	Properties, Value Handle, Characteristic UUID	Read only, no authentication, no authorization

«Characteristics» PROPERTIES

- Properties for Characteristic Value
  - Broadcast
  - Read
  - Command
  - Write
  - Notify
  - Indicate
  - Signed Command
  - Extended Properties

CONTROL-POINT CHARACTERISTICS

- Characteristics that are only:
  - Command, Write, Notify or Indicate are called “Control-Point Characteristics”

- Cannot read Control-Point Characteristics
  - They expose no state
  - Can represent “instantaneous” state

«Characteristic» HANDLE AND TYPE

- Handle for Characteristic Value not incremental to Declaration

- Type is repeated for optimized searches
  - Read By Type «Device Name»
  - Read By Type «Characteristic»

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

CHARACTERISTIC DESCRIPTORS

- «Characteristic Extended Properties»
  - Additional properties (that didn't fit in «Characteristic»)
- Bit: Reliable Write
  - Can "prepare / execute write"
- Bit: Writable Descriptors
  - Can write descriptors (when allowed)

CHARACTERISTIC DESCRIPTORS

- «Characteristic User Description»
  - UTF-8 string

- Description of this characteristic
  - Typically written by user to label a characteristic
  - May require authentication / authorization to write

- Max of one User Description per Characteristic

CHARACTERISTIC DESCRIPTORS

- «Client Characteristic Configuration»
  - Notification / Indication configuration
- Allows each client to turn on / off notifications / indications
  - Does not define when these are sent
  - Just that they are sent
- Different value for each client that is “bonded” with device
  - May require authentication / authorization to write

CHARACTERISTIC DESCRIPTORS

- «Server Characteristic Configuration»
  - Broadcast configuration

- Allows any client to turn on / off broadcast
  - Does not define when these are sent
  - Just that they are sent

- One value for all clients
  - May require authentication / authorization to write

CHARACTERISTIC DESCRIPTORS

- «Characteristic Presentation Format»
  - Defines how the characteristic value if formatted
  - Used to “present” value to user via a “Generic Client”

- Fields include:
  - Format (boolean, uint8, sint16, float32, utf8s, etc…)
  - Exponent (multiply value by $10^{Exponent}$)
  - Unit (SI Units / Derived Units)
  - Name Space (Who allocated Description: Bluetooth, Continua)
  - Description (Above, Below, Armpit, Ear, Inside, Outside, etc…)

CHARACTERISTIC DESCRIPTORS

- «Characteristic Aggregate»
  - List of characteristic presentation formats for each part of the aggregated value
- «Longitude», «Latitude», «Elevation»
- «3D Position»
  - Aggregate Longitude / Latitude / Elevation

GATT PROCEDURES

Procedure	Sub-Procedures
Server Configuration	Exchange MTU
Primary Service Discovery	Discovery All Primary Service Discover Primary Service by Service UUID
Relationship Discovery	Find Included Services
Characteristic Discovery	Discover All Characteristics of a Service Discover Characteristics by UUID
Characteristic Descriptor Discovery	Discover All Characteristic Descriptors
Characteristic Value Read	Read Characteristic Value Read Using Characteristic UUID Read Long Characteristic Values Read Multiple Characteristic Values

GATT PROCEDURES

Procedure	Sub-Procedures
Characteristic Value Write	Write Without Response Write Without Response With Authentication Write Characteristic Value Write Long Characteristic Values Reliable Writes
Characteristic Value Notifications	Notifications
Characteristic Value Indications	Indications
Characteristic Descriptors	Read Characteristic Descriptors Read Long Characteristic Descriptors Write Characteristic Descriptors Write Long Characteristic Desc

GATT CHARACTERISTICS

- GATT defines it own «GATT» Service, and characteristics
  - «Service Changed»

«Service Changed» CHARACTERISTIC

- Control-Point Characteristic
  - Notified when client uses attributes on a server
  - After server has change attribute handles
  - Or after server has added or removed services
- Client must perform “Service Discovery” procedure
  - Re-establish services / characteristics
- Allows the client to cache attribute handles and to determine when the attribute handle cache is no longer valid
- Not required if no services or characteristics can be added or removed, i.e. attributes handles never change

GATT SUMMARY

- Defines grouping of attributes
  - Services

    Include
  - Characteristics
    - Descriptors

- Grouping Attributes are called Descriptors
  - «Primary Service», «Secondary Service», «Characteristic»

Questions?

THANK YOU!