Telephone Bearer Service

***Bluetooth*® Service Specification**

- **Revision:** v1.0
- **Revision Date:** 2021-03-09
- **Group Prepared By:** Generic Audio Working Group

Abstract:

This service exposes a telephone call control interface and telephone call control status for bearers on devices that can make and receive phone calls.

Revision History

Revision Number	Date	Comments
v1.0	2021-03-09	Adopted by the Bluetooth SIG Board of Directors.

Contributors

Name	Company
Tim Reilly	Bose Corporation
Robin Heydon	Qualcomm, Inc.
Michael Rougeux	Bose Corporation
Chris Church	Qualcomm, Inc.
Jonathan Tanner	Qualcomm, Inc.
Andrew Estrada	Sony Corporation
Masahiko Seki	Sony Corporation
Marcel Holtmann	Intel Corporation
Nick Hunn	GN Resound
Scott Walsh	Plantronics Inc.
Luiz Von Dentz	Intel Corporation
Oren Haggai	Intel Corporation
Rasmus Abildgren	Bose Corporation
Leif-Alexandre Aschehoug	Nordic Semiconductor ASA
Frank Yerrace	Microsoft Corporation

Contents

1 Introduction

1.1 Conformance

If conformance to this specification is claimed, all capabilities indicated as mandatory for this specification shall be supported in the specified manner (process-mandatory). This also applies for all optional and conditional capabilities for which support is indicated.

1.2 Service dependencies

This service is not dependent upon any other services.

1.3 Bluetooth Core Specification release compatibility

This specification is compatible with any version of the Bluetooth Core Specification that includes the Generic Attribute Profile (GATT).

1.4 GATT sub-procedure requirements

Requirements in this section represent a minimum set of requirements for a server. Other GATT sub-procedures may be used if supported by both the client and server.

Table 1.1 summarizes additional GATT sub-procedure requirements beyond those required by all GATT servers.

Requirements in this section are defined as “Mandatory” (M), “Optional” (O), “Excluded” (X), and “Conditional” (C.*n*). Conditional statements (C.*n*) are listed directly below the table in which they appear.

GATT Sub-Procedure	Requirements
Write Without Response	M
Write Characteristic Value	M
Notifications	M
Read Characteristic Descriptors	M
Write Characteristic Descriptors	M

Table 1.1: Additional GATT sub-procedure requirements

If the server supports characteristic values larger than the minimum ATT_MTU for the Unenhanced ATT bearer, then the server should support the Read Long Characteristic Values GATT sub-procedure if not already required by the Core Specification [5].

The server shall return an ATT Error Response with the application error code Value Changed During Read Long (0x80) as defined in Table 1.2 to any ATT_READ_BLOB_REQ with a non-zero Value Offset if the characteristic value has changed since the last zero offset read response of that characteristic value to that client.

1.5 Transport dependencies

This service uses GATT and therefore has no additional transport dependencies.

Notifications with GATT are considered unreliable when used with an unenhanced Attribute Protocol (ATT) bearer. See Volume 3, Part F, Section 3.3.2 in the Bluetooth Core Specification [5].

An Enhanced ATT bearer can be used for reliability of notifications and can be specified by a higher-layer profile.

1.6 Application error codes

This service defines the ATT Application error codes shown in Table 1.2.

Name	Error Code	Description
Value Changed During Read Long	0x80	A characteristic value has changed while a Read Long Value Characteristic sub-procedure is in progress.

Table 1.2: Application error codes

1.7 Byte transmission order

All characteristics used with this service shall be transmitted with the least significant octet (LSO) first (i.e., little endian). The LSO is identified in the characteristic definitions in [1].

1.8 Language

1.8.1 Language conventions

The Bluetooth SIG has established the following conventions for use of the words ***shall***, ***must***, ***will***, ***should***, ***may***, ***can***, ***is***, and ***note*** in the development of specifications:

shall	is required to – used to define requirements.
must	is used to express: a natural consequence of a previously stated mandatory requirement. OR an indisputable statement of fact (one that is always true regardless of the circumstances).
will	it is true that – only used in statements of fact.
should	is recommended that – used to indicate that among several possibilities one is recommended as particularly suitable, but not required.
may	is permitted to – used to allow options.
can	is able to – used to relate statements in a causal manner.
is	is defined as – used to further explain elements that are previously required or allowed.
note	Used to indicate text that is included for informational purposes only and is not required in order to implement the specification. Each note is clearly designated as a “Note” and set off in a separate paragraph.

For clarity of the definition of those terms, see Core Specification Volume 1, Part E, Section 1.

1.8.2 Reserved for Future Use

Where a field in a packet, Protocol Data Unit (PDU), or other data structure is described as "Reserved for Future Use" (irrespective of whether in uppercase or lowercase), the device creating the structure shall set its value to zero unless otherwise specified. Any device receiving or interpreting the structure shall ignore that field; in particular, it shall not reject the structure because of the value of the field.

Where a field, parameter, or other variable object can take a range of values, and some values are described as "Reserved for Future Use," a device sending the object shall not set the object to those values. A device receiving an object with such a value should reject it, and any data structure containing it, as being erroneous; however, this does not apply in a context where the object is described as being ignored or it is specified to ignore unrecognized values.

When a field value is a bit field, unassigned bits can be marked as Reserved for Future Use and shall be set to 0. Implementations that receive a message that contains a Reserved for Future Use bit that is set to 1 shall process the message as if that bit was set to 0, except where specified otherwise.

The acronym RFU is equivalent to Reserved for Future Use.

1.8.3 Prohibited

When a field value is an enumeration, unassigned values can be marked as “Prohibited.” These values shall never be used by an implementation, and any message received that includes a Prohibited value shall be ignored and shall not be processed and shall not be responded to.

Where a field, parameter, or other variable object can take a range of values, and some values are described as “Prohibited,” devices shall not set the object to any of those Prohibited values. A device receiving an object with such a value should reject it, and any data structure containing it, as being erroneous.

“Prohibited” is never abbreviated.

1.9 Terminology

Table 1.3 defines terms that are needed to understand this service.

Term	Definition
Caller ID	The string of letters and/or numbers used when originating a call, not including the Uniform Resource Identifier (URI) scheme.
Client	The device(s) that connects to the Telephone Bearer Service (TBS) and accesses the characteristics on the TBS.
Inband ringtone	A sound generated by the server that indicates an incoming call.
Local	The “near end” of a phone call.
Local Hold	A client or server device puts a call on hold from the perspective of the local device. This could be considered as a traditional hold.
Local Retrieve	A call that is in the Locally Held state (controlled by the phone or the “local” device) is removed from Local Hold.
Remote	The “far end” of a phone call.
Remote Hold	A remote party puts a call on hold. In traditional cell phone networks, there is no indication given to the local party that they have been put on hold; the call on hold still acts like an active call. However, in Voice over Internet Protocol (VoIP) apps, for example, the network provides an indication back to the local party that the remote party put the call on hold.

Term	Definition
Server	The device that contains the TBS.
Silent mode	A configuration of the device that contains the server that disables ringtones and other tones on the device.

Table 1.3: Terminology

2 Service

2.1 Declaration

This specification describes two services: TBS and Generic Telephone Bearer Service (GTBS).

A device can have multiple, logically separate, internal telephone bearers.

TBS shall be a «Primary Service» and the service universally unique identifier (UUID) shall be set to «Telephone Bearer», as defined in [1]. There can be zero or more instances of TBS.

GTBS shall be a «Primary Service» and the service UUID shall be set to «Generic Telephone Bearer», as defined in [1]. There shall be one instance of GTBS.

Unless otherwise stated in this specification, the behavior of TBS and GTBS is identical.

2.2 Included services

No additional services are included in this service.

2.3 Behavior

TBS is instantiated on devices that can make and/or receive phone calls. These devices include cell phones; tablets; personal computers (PCs); laptops; wearable devices such as smart watches and smart speakers/displays; and conference room equipment (video or audio only).

2.3.1 Telephone Bearer Service and Generic Telephone Bearer Service

TBS and GTBS both expose characteristics that provide status and control of the telephone bearers.

A device can have multiple, logically separate, telephone bearers. A TBS instance provides status and control for a specific telephone bearer within the device. A device implements TBS instances when the device wants to allow clients to directly access and control the characteristics of separate internal telephone bearer entities.

GTBS provides a single point of access and exposes a representation of its internal telephone bearers into a single telephone bearer. This service provides telephone status and control of the device as a single unit with a single set of characteristics. It is left up to the implementation to determine what telephone bearer a characteristic of GTBS represents at any time. There is no specified manner of representing a characteristic from each individual TBS that resides on the device to the same characteristic of the GTBS.

For example, if there is more than one TBS on a device and each has a unique telephone bearer name (e.g., Name1 and Name2), the way the GTBS represents the telephone bearer name is left up to the implementation. GTBS is suited for clients that do not need to access or control all the information available on specific telephone bearers.

For example, a device like a phone or PC can have many end-user applications that expose telephony services (e.g., cellular call applications and VoIP applications). The device can treat these applications as unique telephone bearers. The device can implement a separate TBS instance for each telephone bearer application, which allows clients to discover, observe, and control the telephone bearer applications individually. In addition, the device can implement a single GTBS instance to provide a single set of characteristics that provides the status and control of the telephony services on the device, but in a manner left up to the implementation.

Information about the different states of each active telephone call and methods for the client device to request a change to the state of the call based on actions at the client device is also exposed. Each active call has a Call State characteristic and a Call Control Point characteristic that can be described by the state machine, as shown in Figure 2.1. The Call State definition is identical for TBS and GTBS.

GTBS and each individual TBS have their own Call Index number space. There is no requirement that the Call Index that is exposed by TBS matches the Call Index that is exposed by GTBS (even if the same call is exposed by TBS and GTBS).

The transitions in the state machine in Figure 2.1 show both:

- Client-initiated transitions (resulting from writes to the Call Control Point)
- Server-based transactions (resulting from user interaction on the server, internal server actions, or actions taken by a remote party to a phone call)

For descriptions of the states and transitions, see Table 3.6 and Table 3.9.

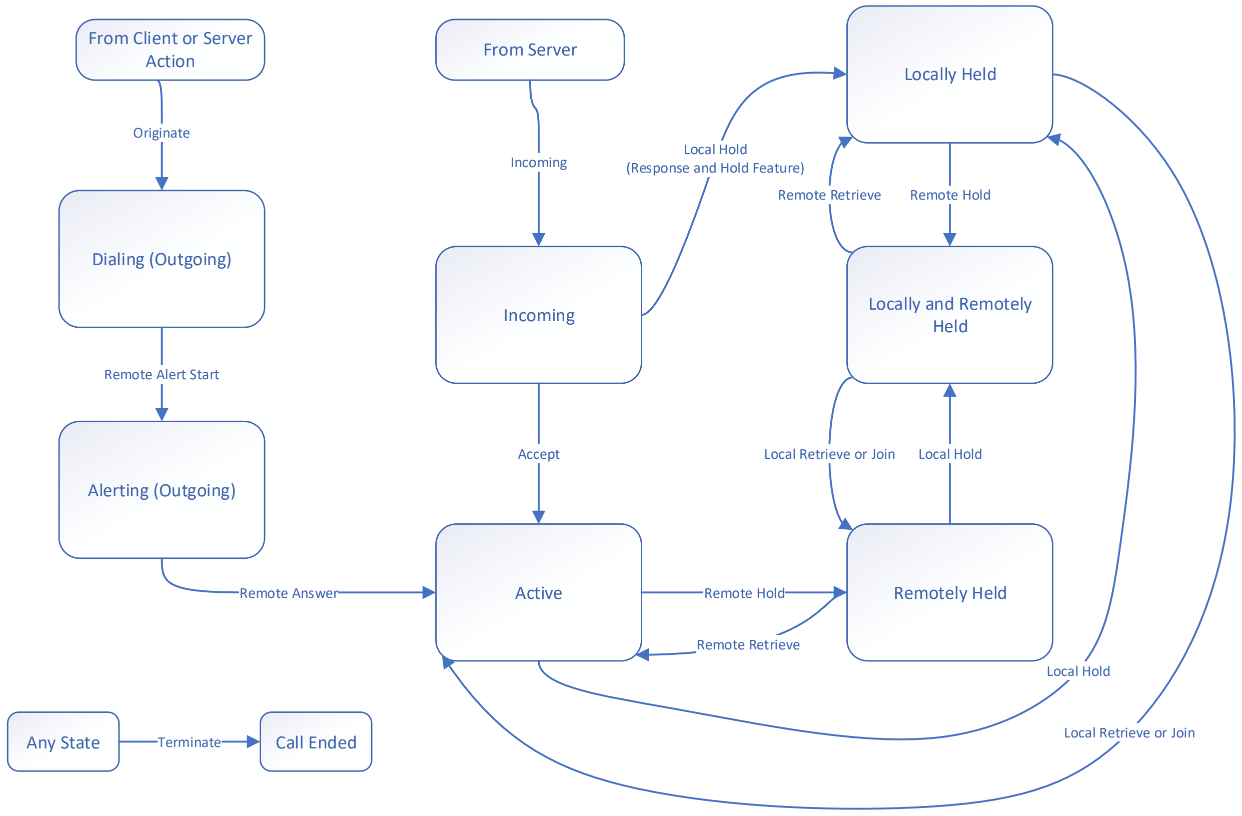

Figure 2.1: State machine describing the Call State characteristic for a single call

2.3.2 Multi-Party Calling

TBS and GTBS provide a way to allow multi-party calling. Because each call maintains its own state machine, individual calls can be placed into states that are used with multi-party calls. Examples are provided for Call 1 and Call 2 in the discussion in the following paragraph.

If a user is in a call (Call 1) and receives a new call (Call 2), then the Call 2 state should be set to Incoming. The user can then take the appropriate actions on each call to achieve the desired result, which include:

1. Place Call 1 on hold and answer Call 2.
2. Answer Call 2 and end Call 1.
3. Keep talking on Call 1 and place Call 2 on hold.
4. Keep talking on Call 1 and reject Call 2 (send to voicemail).
5. Join Call 1 and Call 2 together.

Each result described in this list can be achieved by performing the appropriate actions and having the subsequent state changes based on the state machine of each call illustrated in Figure 2.1.

3 Service characteristics

This section defines the characteristic and descriptor requirements for TBS and GTBS. Multiple instances of TBS can exist on a device, for example, when one telephone bearer is a cellular provider and another telephone bearer is VoIP. To support multiple simultaneous calls, characteristics return arrays of data.

Requirements in this section are defined as “Mandatory” (M), “Optional” (O), “Excluded” (X), and “Conditional” (C.*n*). Conditional statements (C.*n*) are listed directly below the table in which they appear.

Table 3.1 defines the characteristic and descriptor requirements for each instance of TBS and the single instance of GTBS.

Characteristic Name	Requirement	Mandatory Properties	Optional Properties	Security Permissions
Bearer Provider Name	M	Read, Notify	None	Encryption required
Bearer Uniform Caller Identifier (UCI)	M	Read	None	Encryption required
Bearer Technology	M	Read, Notify	None	Encryption required
Bearer URI Schemes Supported List	M	Read	Notify	Encryption required
Bearer Signal Strength	O	Read, Notify	None	Encryption required
Bearer Signal Strength Reporting Interval	C.1	Read, Write, Write Without Response	None	Encryption required
Bearer List Current Calls	M	Read, Notify	None	Encryption required
Content Control ID (CCID)	M	Read	None	Encryption required
Status Flags	M	Read, Notify	None	Encryption required
Incoming Call Target Bearer URI	O	Read, Notify	None	Encryption required
Call State	M	Read, Notify	None	Encryption required
Call Control Point	M	Write, Write Without Response, Notify	None	Encryption required

Characteristic Name	Requirement	Mandatory Properties	Optional Properties	Security Permissions
Call Control Point Optional Opcodes	M	Read	None	Encryption required
Termination Reason	M	Notify	None	Encryption required
Incoming Call	M	Read, Notify	None	Encryption required
Call Friendly Name	O	Read, Notify	None	Encryption required

Table 3.1: TBS and GTBS characteristics

C.1: Mandatory if Bearer Signal Strength is supported, otherwise Excluded.

3.1 Bearer Provider Name

The Bearer Provider Name characteristic returns the friendly name of the provider, for example, a cellular service provider. The characteristic value shall be an 8-bit Unicode Transformation Format (UTF-8) string. The value of this characteristic can change if the provider changes. For example, this can occur when a phone is roaming and changes providers as it roams through different coverage areas.

3.1.1 Bearer Provider Name behavior

The Bearer Provider Name characteristic returns its associated value when read by a client. The characteristic can be configured for notification by using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor.

When the Bearer Provider Name changes (e.g., when a phone is roaming), the characteristic value is updated. If the value of the Client Characteristic Configuration descriptor is configured for notification, then the server shall send a notification to the client when there is a change in the characteristic value. If the characteristic value is longer than (ATT_MTU-3), then the first (ATT_MTU-3) octets shall be included in the notification.

3.2 Bearer UCI

The Bearer UCI characteristic returns the identifier of the telephone bearer, as identified in the UCI table [2].

The “client_id” field of the UCI table is used to populate the Bearer UCI characteristic as a UTF-8 string.

3.2.1 Bearer UCI behavior

The Bearer UCI characteristic returns its associated value when read by a client.

This characteristic provides a common representation of a telephone bearer between the client and server. This is accomplished by obtaining the UCI from the assigned numbers file defined in [2].

The server shall not change this value once instantiated.

3.3 Bearer Technology

The Bearer Technology characteristic returns information about the type of technology that is being used for this telephone bearer. Terms such as 3G, 4G, Long Term Evolution (LTE), and Wi-Fi are examples of values and are defined in the Bluetooth assigned numbers [1].

3.3.1 Bearer Technology behavior

The Bearer Technology characteristic returns its associated value when read by a client. The characteristic can be configured for notification by using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor. When configured for notification, this characteristic can be notified while the client and server are in a connection.

When the technology changes (e.g., coverage goes from 3G to 4G to LTE), the characteristic value is updated. If the value of the Client Characteristic Configuration descriptor is configured for notification, then the server shall send a notification to the client when the characteristic value changes.

This characteristic should be used to populate a user interface (UI). The client should make no assumptions about the behavior or supported features based on the value of this characteristic.

3.4 Bearer URI Schemes Supported List

The Bearer URI Schemes Supported List characteristic returns the schemes that are supported by this telephone bearer.

The characteristic shall be a comma-delimited list of supported schemes expressed as a UTF-8 string. Examples of common URI schemes can be found in [3]. An example characteristic value is the UTF-8 string “tel,sip,skype”.

3.4.1 Bearer URI Schemes Supported List behavior

The Bearer URI Schemes Supported List characteristic returns its associated value when read by a client. A comma-delimited list of supported schemes is returned.

When the supported schemes change, the characteristic value is updated. If the value of the Client Characteristic Configuration descriptor is configured for notification, then the server shall send a notification to the client when there is a change in the characteristic value. If the characteristic value is longer than (ATT_MTU-3), then the first (ATT_MTU-3) octets shall be included in the notification.

3.5 Bearer Signal Strength

The Bearer Signal Strength characteristic is 1 octet, returns the level of the signal of the telephone bearer, and can hold a value from 0 to 100, or 255. Values 101 to 254 are RFU. A value of 0 indicates that there is no service, a value of 100 indicates the maximum signal strength, and a value of 255 indicates that the signal strength is unavailable or has no meaning for this particular bearer. The meaning of the values between 1 to 99 is left up to the implementation.

3.5.1 Bearer Signal Strength behavior

The Bearer Signal Strength characteristic returns its associated value when read by a client. The characteristic can be configured for notification using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor. When configured for notification, this characteristic can be notified while the client and server are in a connection.

When the signal strength changes, the characteristic value is updated. If the value of the Client Characteristic Configuration descriptor is configured for notification, then the server shall send a notification to the client when there is a change in the characteristic value. The reporting interval is based on the value described by the Bearer Signal Strength Reporting Interval characteristic. The reporting interval is described in Section 3.6.

3.6 Bearer Signal Strength Reporting Interval

The Bearer Signal Strength Reporting Interval characteristic is 1 octet and returns the reporting interval expressed in seconds. The valid range of values for the Reporting Interval is 0 to 255. If the interval is set to 0, the signal strength is reported as soon as the signal strength changes. The server should set a default value of 0 to preclude any unwanted signal strength notifications to the client. No reporting occurs unless the Bearer Signal Strength characteristic is configured for notification.

3.6.1 Bearer Signal Strength Reporting Interval behavior

The Bearer Signal Strength Reporting Interval characteristic returns its associated value when read by a client.

When the Bearer Signal Strength Reporting Interval characteristic is written, the server shall use the new reporting interval when reporting the Bearer Signal Strength.

The Bearer Signal Strength characteristic value shall be notified to the client if the Reporting Interval is a non-zero value and if the signal strength has changed since the last notification. The minimum time between successive notifications shall be at least the Reporting Interval value. It is left up to the implementation to determine the exact time that the notification of the Bearer Signal Strength is sent, but the notifications shall not occur more frequently than the interval set.

3.7 Bearer List Current Calls

The Bearer List Current Calls characteristic returns a list of current calls. If no calls are available, the list is empty, and a read of this characteristic shall return 0 octets. The list of current calls is built from other characteristics from TBS and is provided in this format to offer a way to get a snapshot of the current call information.

Each list item in the list of calls shall have the structure shown in Table 3.2.

The List_Item_Length field is the total length of the item, excluding the octet for the List_Item_Length field itself.

List_Item_Length[i] (1 octet)	Call_Index[i] (1 octet)	Call_State[i] (1 octet)	Call_Flags[i] (1 octet)	Call_URI[i] (variable)

Table 3.2: Bearer List Current Calls characteristic format

The fields provided in Table 3.2 are described in more detail in the following sections:

- Call_Index: Section 3.11
- Call_State: Section 3.11
- Call_Flags: Section 3.11
- Incoming_Call_URI: Section 3.15

- Outgoing_Call_URI used in the Originate Call Control Point write: Section 3.12.1.5.

3.7.1 Bearer List Current Calls behavior

The Bearer List Current Calls characteristic returns its associated value when read by a client. The characteristic can be configured for notification using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor. When configured for notification, this characteristic can be notified while the client and server are in a connection.

When any of the characteristics that are used to populate the fields of the Bearer List Current Calls characteristic change, the characteristic value is updated. If the value of the Client Characteristic Configuration descriptor is configured for notification, then the server shall send a notification to the client when there is a change in the characteristic value. If the characteristic value is longer than (ATT_MTU-3), then the first (ATT_MTU-3) octets shall be included in the notification.

The Incoming Call URI or Outgoing Call URI is used to populate the URI field, depending on whether the call is an incoming or outgoing call.

3.8 Content Control ID

The server shall expose the CCID characteristic as defined in [4].

3.8.1 Content Control ID behavior

The CCID characteristic returns its associated value when read by a client.

The CCID characteristic in each instance of the service shall persist across connections until the handle range is affected by a service change.

There shall only be one instance of the CCID characteristic in an instance of TBS.

3.9 Status Flags

The Status Flags characteristic is 2 octets and returns the current status of features that the server device supports.

3.9.1 Status Flags behavior

The Status Flags characteristic returns its associated value when read by a client. The characteristic can be configured for notification using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor. When configured for notification, this characteristic can be notified while the client and server are in a connection.

Table 3.3 describes the definitions of the bit fields that are used by the server to indicate various features of the server that are enabled or disabled. The RFU bits shall be set to 0. If notifications are enabled, the Status Flags characteristic shall be notified on a change.

Bit	**Description**
Bit 0	Inband ringtone 0=inband ringtone disabled 1=inband ringtone enabled
Bit 1	Silent mode 0 = Server is not in silent mode 1 = Server is in silent mode
Bit 2–15	RFU

Table 3.3: Status Flags characteristic bit definitions

3.10 Incoming Call Target Bearer URI

The Incoming Call Target Bearer URI characteristic returns the Call Index followed by the URI of the target of an incoming call. An example of the URI is "tel:+15025551212" and is formatted as a UTF-8 string. The format of the characteristic is defined in Table 3.4.

Call_Index (1 octet)	Incoming_Call_Target_URI (variable)

Table 3.4: Incoming Call Target Bearer URI characteristic format

3.10.1 Incoming Call Target Bearer URI behavior

The Incoming Call Target Bearer URI characteristic returns its associated value when read by a client. The characteristic can be configured for notification by using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor. When configured for notification, this characteristic shall be notified while the client and server are in a connection. If the characteristic value is longer than (ATT_MTU-3), then the first (ATT_MTU-3) octets shall be included in the notification.

When an incoming call arrives at the telephone bearer, the Incoming_Call_Target_Bearer_URI field is populated with the URI scheme followed by the identity of the target of the call and notified to the client if notifications are enabled.

3.11 Call State

The Call State characteristic returns an array of members, each 3 octets, and represents an abstraction of the different states that telephone calls are in. The format of the characteristic is described in Table 3.5.

Call_Index[i] (1 octet)	State[i] (1 octet)	Call_Flags[i] (1 octet)

Table 3.5: Call State characteristic format

Each telephone call is identified by a Call_Index value that is unique per service. The Call_Index octet is a number chosen by the server to identify an active call. The Call_Index value exposed in this Call_State characteristic is used throughout this service to identify calls.

Each telephone call is in one of the listed states. Table 3.6 lists the states and provides a short description of each.

State	Name	Description
0x00	Incoming	A remote party is calling (incoming call).
0x01	Dialing	The process to call the remote party has started on the server, but the remote party is not being alerted (outgoing call).
0x02	Alerting	A remote party is being alerted (outgoing call).
0x03	Active	The call is in an active conversation.
0x04	Locally Held	The call is connected but held locally. “Locally Held” implies that either the server or the client can affect the state.
0x05	Remotely Held	The call is connected but held remotely. “Remotely Held” means that the state is controlled by the remote party of a call.
0x06	Locally and Remotely Held	The call is connected but held both locally and remotely.
0x07–0xFF	–	RFU

Table 3.6: Call State characteristic possible states

Each telephone call has a Call_Flags field. Table 3.7 lists the definitions of bits of this field.

Bit	Description
Bit 0	Incoming/Outgoing 0 = Call is an incoming call 1 = Call is an outgoing call
Bit 1	Information withheld by server 0 = Not withheld 1 = Withheld
Bit 2	Information withheld by network 0 = Provided by network 1 = Withheld by network
Bits 3-7	RFU

Table 3.7: Call_Flags bit definitions

3.11.1 Call State behavior

The Call State characteristic returns its array of associated values when read by a client.

The Call State characteristic can be configured for notification by using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor. When configured for notification, this characteristic can be notified while the client and server are in a connection.

When the service instantiates, the Call_Index shall be a number from 1 to 255. With each call, the Call_Index should increment by 1. When the Call_Index reaches 255, it should roll over to 1. The Call_Index of 0 is reserved. The Call_Index shall remain unique for the life of the call.

When a server first allocates a Call_Index to be included in the Call State list, the Call State characteristic should be notified before any other characteristic that includes the Call_Index.

When the state of the call changes, the server shall update the Call State characteristic with the state. The Incoming/Outgoing bit of the Call_Flags field is set to 1 if the call is an outgoing call and to 0 if the call is an incoming call.

When a server allocates a Call_Index, the server may withhold the URI and friendly name fields of characteristics containing fields with a URI or friendly name from the client. This will be indicated by setting one or both of the Withheld by Server or Withheld by Network bits in Call_Flags. Bit 1 shall be set if a policy or configuration of the server requires it. Bit 2 shall be set if the network cannot provide it. If the fields are withheld for any reason, they may be NULL, empty, or zero-sized, however, the server may also add a field such as ‘Withheld’ or ‘Unknown Caller’.

If the value of the Client Characteristic Configuration descriptor is configured for notification, then the server shall send a notification to the client when there is a change in the characteristic value. If the characteristic value is longer than (ATT_MTU-3), then the first (ATT_MTU-3) octets shall be included in the notification.

3.12 Call Control Point

The Call Control Point characteristic varies in length, depending on which opcode is used. The first octet is an enumerated value known as the opcode. The parameter fields that follow the opcode have different content based on the opcode. The format and content are described in Table 3.8 and Table 3.9. Optional opcodes are listed in the Call Control Point Optional Opcodes characteristic (see Section 3.13). All other opcodes are mandatory.

Opcode (1 octet)	Parameter (variable)

Table 3.8: Call Control Point characteristic format

Opcode Name	Opcode	Parameter	Notes
Accept	0x00	Call_Index (1 octet)	Answer the incoming call identified by Call_Index.
Terminate	0x01	Call_Index (1 octet)	End the active, alerting (outgoing), dialing (outgoing), incoming, or held (locally or remotely) call identified by Call_Index.
Local Hold	0x02	Call_Index (1 octet)	Place the active or incoming call identified by Call_Index on local hold.
Local Retrieve	0x03	Call_Index (1 octet)	Move a locally held call identified by Call_Index to an active call. Move a locally and remotely held call to a remotely held call.
Originate	0x04	URI (URI length octets)	Initiate a call to the remote party identified by the URI.
Join	0x05	List_of_Call_Indexes (Number of Call Indexes octets)	Put calls in the list that are not in the Remotely Held state to the active state and join the calls. Any calls in the Remotely Held or Locally and Remotely Held states move to the Remotely Held state and are joined with the other calls.
–	0x06–0xFF	RFU	–

Table 3.9: Call Control Point characteristic opcodes and parameters

3.12.1 Call Control Point behavior

When the Call Control Point characteristic is written, the server shall perform the behavior as defined in Sections 3.12.1.1, 3.12.1.2, 3.12.1.3, 3.12.1.4, 3.12.1.5, and 3.12.1.6.

The parameter fields are populated by the client with the value of the Call_Index of the Call State characteristic for the call that is being controlled, the URI information for an outgoing call, or a list of Call Indexes to join.

The Call Control Point characteristic can be configured for notification by using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor. When configured for notifications, this characteristic can be notified while the client and server are in a connection. See Section 3.12.2 for a description of the Call Control Point Notification.

3.12.1.1 Accept

The Accept opcode only has meaning when the Call State characteristic is Incoming.

When Accept is written as the Control Point opcode, the server shall start the necessary procedures for the given telephone bearer to accept the call, and the Call State transitions to Active.

If Accept is written to the Call Control Point, any other calls that are in the Active state should be moved to the Locally Held state. If any calls are in the Remotely Held state, they should be moved to the Locally and Remotely Held state. To have multiple calls active at the same time, the Join opcode should be used.

The Parameter field contains the Call Index to be accepted.

3.12.1.2 Terminate

The Terminate opcode has meaning when the Call State is in any state.

If the Call State is in any state, when Terminate is written as the Control Point opcode, the call is terminated, and the Call State characteristic associated with the terminated call shall be removed from the Call State list.

When a call is terminated, the server shall populate the Termination Reason characteristic with the Call Index and Reason Code.

The Parameter field contains the Call Index to be terminated.

3.12.1.3 Local Hold

The Local Hold opcode only has meaning when the Call State is Incoming, Active, or Remotely Held.

If the Call State is Incoming or Active, when Local Hold is written as the Control Point opcode, the Call State transitions to Locally Held.

If the Call State is Remotely Held, when Local Hold is written as the Control Point opcode, the Call State transitions to Locally and Remotely Held.

The Parameter field contains the Call Index to be held locally.

3.12.1.4 Local Retrieve

The Local Retrieve opcode only has meaning when the Call State is set to Locally Held or Locally and Remotely Held.

If the Call State is Locally Held, when Local Retrieve is written as the Control Point opcode, the Call State transitions to Active. Any calls with a Call State of Active are expected to transition to the Locally Held state.

If the Call State is Locally and Remotely Held when Local Retrieve is written as the Control Point opcode, the Call State transitions to Remotely Held. Any calls with a Call State of Active transition to the Locally Held state. Any calls with a Call State of Remotely Held transition to the Locally and Remotely Held state.

The Retrieve opcode should not be used to join calls. The Join opcode is used to join calls.

The Parameter field contains the Call Index to be retrieved locally.

3.12.1.5 Originate

The Originate opcode has meaning at any time.

When Originate is written as the Control Point opcode, the Call State transitions to Dialing (Outgoing) with a new Call Index. If there are any active calls when the Originate opcode is accepted, the calls should be moved to the Locally Held state. Based on the implementation, when alerting starts, the Call State transitions to Alerting. When the call is answered on the remote end, the Call State transitions to Active.

The Parameter field contains the URI of the outgoing call. The URI is comprised of the URI scheme followed by the Caller ID (this could be a telephone number or username). For example, if the URI scheme is “skype” and the Caller ID (username) is “xyz,” the resultant URI is “skype:xyz” formatted as a UTF-8 string. If the URI scheme is “tel” and the Caller ID is “+15025551212,” the resultant URI is “tel:+15025551212,” formatted as a UTF-8 string.

3.12.1.6 Join

This opcode only has meaning when two or more calls are in any state, with the exception of Incoming.

If the server supports the joining of calls, when Join is written as the Control Point opcode, the calls included in the write to the characteristic are expected to be joined, and the Call State of each call is handled as follows:

- If the Call State of the call is Active, the Call State shall remain Active.
- If the Call State of the call is Locally Held, the Call State transitions to Active.
- If the Call State of the call is Remotely Held, the Call State remains Remotely Held.
- If the Call State of the call is Locally and Remotely Held, the Call State transitions to Remotely Held.
- If the Call State of the call is Alerting or Dialing, the Call State remains in that state. It is left up to the server implementation on how the call is joined after transitioning to the Active state with a remote answer.
- If the Call State of any call NOT in the list of calls to be joined is Active, the Call State for each of these calls transitions to Locally Held.

If the network does not support joining of calls, or if any Call Index in the provided list has a Call State of Incoming, or less than two calls are in the list provided, the OPERATION NOT POSSIBLE Result Code shall be provided in the Control Point Notification to the client. No calls shall be joined, and the Call State of calls in the list provided in the Join command shall not change state because of the Join opcode being written.

The Parameter field contains the list of Call Indexes to be joined.

3.12.2 Call Control Point Notification

Table 3.10 shows the required structure of the Call Control Point Notification. The Call Control Point Notification is sent from the server to the client after each Call Control Point opcode write. If the requested opcode procedure was successfully completed and the opcode was not Originate or Join, then the Result Code of the Call Control Point Notification shall be set to SUCCESS and the Call_Index field shall be set to the provided Call Index. If the requested opcode procedure was not successfully completed, the Result Code shall be set to a value from Table 3.11 and the Call_Index field shall be set to 0.

For the Originate opcode, if the opcode procedure was successfully completed, then the Result Code of the Call Control Point Notification shall be set to SUCCESS, and the Call Index field shall be set to the Call Index assigned to the call.

For the Join opcode, if the opcode procedure was successfully completed, then the Result Code of the Call Control Point Notification shall be set to SUCCESS, and the Call_Index field shall be set to the first Call Index in the list of provided Call Indexes.

If an invalid opcode is received, the Result Code shall be set to OPCODE NOT SUPPORTED and the Call_Index field shall be set to 0.

Requested Opcode (1 octet)	Call Index (1 octet)	Result Code (1 octet)

Table 3.10: Call Control Point Notification format

Result Code	Definition	Description
0x00	SUCCESS	The opcode write was successful.
0x01	OPCODE NOT SUPPORTED	An invalid opcode was used for the Call Control Point write.
0x02	OPERATION NOT POSSIBLE	The requested operation cannot be completed.
0x03	INVALID CALL INDEX	The Call Index used for the Call Control Point write is invalid.
0x04	STATE MISMATCH	The opcode written to the Call Control Point was received when the current Call State for the Call Index was not in the expected state.
0x05	LACK OF RESOURCES	Lack of internal resources to complete the requested action.
0x06	INVALID OUTGOING URI	The Outgoing URI is incorrect or invalid when an Originate opcode is sent.
0x07–0xFF	–	RFU.

Table 3.11: Call Control Point Notification Result Codes

3.13 Call Control Point Optional Opcodes

The Call Control Point Optional Opcodes characteristic is 2 octets long and describes the optional opcodes of the Call Control Point. The bits are defined in Table 3.12. If the optional opcode is supported by the server, the associated bit shall be set to 1; otherwise, the bit shall be set to 0. RFU bits in the Call Control Point Optional Opcodes characteristic shall be set to 0.

Bit	Description
Bit 0	Local Hold 0 = Local Hold and Local Retrieve Call Control Point Opcodes not supported 1 = Local Hold and Local Retrieve Call Control Point Opcodes supported
Bit 1	Join 0 = Join Call Control Point Opcode not supported 1 = Join Call Control Point Opcode supported
Bits 2-15	Reserved for Future Use

Table 3.12: Call Control Point Optional Opcodes characteristic bit definitions

3.13.1 Call Control Point Optional Opcodes behavior

When read, the Call Control Point Optional Opcodes characteristic returns a value that is used by a client to determine which optional Call Control Point opcodes are supported by the server. The bits of the Call Control Point Optional Opcodes characteristic are static during a connection.

3.14 Termination Reason

The Termination Reason characteristic is 2 octets, with the first octet being the Call Index of the call that was terminated and the second octet containing a Reason Code that describes the way a call was terminated from the remote end or from the server. The format of the characteristic is shown in Table 3.13 and the Reason Codes are defined in Table 3.14.

Call_Index (1 octet)	Reason_Code (1 octet)

Table 3.13: Termination Reason characteristic format

Reason Code	Reason
0x00	The URI value used to originate a call was formed improperly.
0x01	The call failed.
0x02	The remote party ended the call.
0x03	The call ended from the server.
0x04	The line was busy.
0x05	Network congestion.
0x06	The client terminated the call.
0x07	No service.
0x08	No answer.

Reason Code	Reason
0x09	Unspecified.
0x0A–0xFF	RFU.

Table 3.14: Termination Reason Codes

3.14.1 Termination Reason behavior

The Termination Reason characteristic can be configured for notification by using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor. When configured for notification, this characteristic can be notified while the client and server are in a connection.

When the call is ended, the server shall populate the Termination Reason characteristic. If the value of the Client Characteristic Configuration descriptor is configured for notification, then the server shall send a notification to the client when there is a change in the characteristic value.

3.15 Incoming Call

The Incoming Call characteristic returns the Call Index and URI of the incoming caller. The URI of the incoming call is comprised of the URI scheme followed by the Caller ID of the call as a UTF-8 string. The format of the characteristic is shown in Table 3.15.

Call_Index (1 octet)	URI (variable)

Table 3.15: Incoming Call characteristic format

3.15.1 Incoming Call behavior

The Incoming Call characteristic returns its associated value when read by a client.

The Incoming Call characteristic can be configured for notification by using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor. When configured for notification, this characteristic can be notified while the client and server are in a connection.

This characteristic is populated by the server when a Call State member is created and in the Incoming state. If the value of the Client Characteristic Configuration descriptor is configured for notification, then the server shall send a notification to the client when there is a change in the characteristic value. If the characteristic value is longer than (ATT_MTU-3), then the first (ATT_MTU-3) octets shall be included in the notification.

3.16 Call Friendly Name

The Call Friendly Name characteristic returns the Call Index and friendly name (as determined by the server) of the incoming or outgoing call. The Friendly_Name field is a UTF-8 string. The format of the characteristic is shown in Table 3.16.

Call_Index (1 octet)	Friendly_Name (variable)

Table 3.16: Call Friendly Name characteristic format

3.16.1 Call Friendly Name behavior

The Call Friendly Name characteristic returns its associated value when read by a client.

The Call Friendly Name characteristic can be configured for notification using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor. When configured for notification, this characteristic can be notified while the client and server are in a connection.

This characteristic is populated by the server when there is an incoming or outgoing call and a Call State member is created. This characteristic should be used to populate a UI or spoken to the user. If the friendly name cannot be determined, the server should populate the field with a string that indicates to a user that the friendly name is not known. If the value of the Client Characteristic Configuration descriptor is configured for notification, then the server shall send a notification to the client when there is a change in the characteristic value. If the characteristic value is longer than (ATT_MTU-3), then the first (ATT_MTU-3) octets shall be included in the notification.

4 SDP interoperability

For services that are exposed over Basic Rate/Enhanced Data Rate (BR/EDR), include a Service Discovery Protocol (SDP) Record table, as shown in Table 4.1.

Requirements in this section are defined as “Mandatory” (M), “Optional” (O), “Excluded” (X), and “Conditional” (C.*n*). Conditional statements (C.*n*) are listed directly below the table in which they appear.

Item	Definition	Type	Value	Status
Service Class ID List	–	–	–	M
Service Class #0	–	UUID	«Telephone Bearer»	M
Service Class ID List	–	–	–	M
Service Class #0	–	UUID	«Generic Telephone Bearer»	M
Protocol Descriptor List	–	Data Element Sequence	–	M
Protocol #0	–	UUID	«L2CAP»	M
Parameter #0 for Protocol #0	Protocol/Service Multiplexer (PSM)	Uint16	PSM = ATT	M
Protocol #1	–	UUID	«ATT»	M
Additional Protocol Descriptor List	–	Data Element Sequence	–	C.1
Protocol Descriptor List	–	Data Element Sequence	–	C.1
Protocol #0	–	UUID	«L2CAP»	C.1
Parameter #0 for Protocol #0	PSM	Uint16	PSM = EATT	C.1
Protocol #1	–	UUID	«ATT»	C.1
BrowseGroupList	–	–	PublicBrowseRoot Other browse UUIDs may also be included in the list.	M

Table 4.1: SDP Record table

C.1: Mandatory if EATT is supported, otherwise Excluded.

5 Acronyms and abbreviations

Acronym/Abbreviation	Meaning
ATT	Attribute Protocol
BR/EDR	Basic Rate/Enhanced Data Rate
CCID	Content Control ID
EATT	Enhanced Attribute Protocol
GATT	Generic Attribute Profile
GTBS	Generic Telephone Bearer Service
IANA	Internet Assigned Numbers Authority
L2CAP	Logical Link Control and Adaptation Protocol
LSO	least significant octet
LTE	Long Term Evolution
PC	personal computer
PDU	Protocol Data Unit
PSM	Protocol/Service Multiplexer
RFU	Reserved for Future Use
SDP	Service Discovery Protocol
TBS	Telephone Bearer Service
UCI	Uniform Caller Identifier
UI	user interface
URI	Uniform Resource Identifier
UTF-8	8-bit Unicode Transformation Format
UUID	universally unique identifier
VoIP	Voice over Internet Protocol
Wi-Fi	Wireless Fidelity

Table 5.1: Acronyms and abbreviations

6 References

[1] Bluetooth SIG Assigned Numbers, https://www.bluetooth.com/specifications/assigned-numbers

[2] Bluetooth Uniform Caller Identifiers (UCI) table, https://www.bluetooth.com/specifications/assigned-numbers/uniform-caller-identifiers

[3] Internet Assigned Numbers Authority (IANA) URI Schemes, https://iana.org/assignments/uri-schemes/uri-schemes.xhtml

[4] GATT Specification Supplement, Version 4 or later

[5] Bluetooth Core Specification, Version 5.2 or later