

CoRE
Internet-Draft
Intended status: Standards Track
Expires: 24 October 2026

C. Ams端 ss
22 April 2026

CoAP over GATT (Bluetooth Low Energy Generic Attributes)
draft-ietf-core-coap-over-gatt-00

Abstract

Interaction from computers and cell phones to constrained devices is limited by the different network technologies used, and by the available APIs. This document describes a transport for the Constrained Application Protocol (CoAP) that uses Bluetooth GATT (Generic Attribute Profile) and its use cases.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Constrained RESTful Environments Working Group mailing list (core@ietf.org), which is archived at <https://mailarchive.ietf.org/arch/browse/core/>.

Source for this draft and an issue tracker can be found at <https://github.com/core-wg/coap-over-gatt>.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 24 October 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
1.1. Application example	3
1.2. Alternatives	4
2. Terminology	5
3. Protocol description	5
3.1. Boundary conditions: GATT properties	5
3.2. Requests and responses	6
3.2.1. Message sub-layer	6
3.2.2. Minimal use of the message sub-layer	7
3.2.3. Using the message sub-layer	8
3.2.4. Message deduplication	9
3.2.5. Requests and responses	9
3.2.6. Fragmentation	10
3.2.7. Multiple characteristics	10
3.2.8. Communication example	11
3.3. Addresses	11
3.3.1. Use with persistent addresses	12
4. Further development	13
4.1. Compression and reinterpretation of non-CoAP characteristics	13
4.2. Additional use of advertisements	13
4.3. Protocol details	14
5. IANA considerations	15
5.1. ble.arpa	15
6. Security considerations	16
7. References	16
7.1. Normative References	16
7.2. Informative References	16
Appendix A. Change log	18
Author's Address	20

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] can be used with different network and transport technologies, for example UDP on 6LoWPAN networks.

Not all those network technologies are available at end user devices in the vicinity of the constrained devices, which inhibits direct communication and necessitates the use of gateway devices or cloud services. In particular, 6LoWPAN is not available at all in typical end user devices, and while 6LoWPAN-over-BLE (IPSP, the Internet Protocol Support Profile of Bluetooth Low Energy (BLE), [RFC7668]) might be compatible from a radio point of view, many operating systems or platforms lack support for it, especially in a user-accessible way.

As a workaround to access constrained CoAP devices from end user devices, this document describes a way encapsulate generic CoAP exchanges in Bluetooth GATT (Generic Attribute Profile). This is explicitly not designed as means of communication between two devices in full control of themselves -- those should rather build an IP based network and transport CoAP as originally specified. It is intended as a means for an application to escape the limitations of its environment, with a special focus on web applications that use the Web Bluetooth [webbluetooth]. In that, it is similar to CoAP-over-WebSockets [RFC8323]. GATT, which has read and write semantics, is not a perfect match for CoAP's request/response semantics; this specification bridges the gap in order to make CoAP transportable over what is sometimes the only available protocol.

1.1. Application example

Consider a network of home automation light bulbs and switches, which internally uses CoAP on a 6LoWPAN network and whose basic pairing configuration can be done without additional electronic devices.

Without CoAP-over-GATT, an application that offers advanced configuration requires the use of a dedicated gateway device or a router that is equipped and configured to forward between the 6LoWPAN and the local network. In practice, this is often delivered as a wired gateway device and a custom app.

With CoAP-over-GATT, the light bulbs can advertise themselves via BLE, and the configuration application can run as a web site. The user navigates to that web site, and it asks permission to contact the light bulbs using Web Bluetooth. The web application can then exchange CoAP messages directly with the light bulb, and have it proxy requests to other devices connected in the 6LoWPAN network.

For browsers that do not support Web Bluetooth, the same web application can be packaged into an native application consisting of a proxy process that forwards requests received via CoAP-over-WebSockets on the loopback interface to CoAP-over-GATT, and a browser view that runs the original web application in a configuration to use WebSockets rather than CoAP-over-GATT.

That connection is no replacement when remote control of the system is desired (in which case, again, a router is required that translates 6LoWPAN to the rest of the network), but suffices for many commissioning tasks.

1.2. Alternatives

Several approaches were considered, but considered unsuitable for the intended use cases:

- * CoAP over 6LoWPAN over BLE (BLE IPSP): While this is the natural choice for transporting CoAP over BLE, it is unavailable on typical end user devices. There is no clear path toward how that would be integrated in platforms like Android or iOS, and even if it were, creating a network connection to a nearby device from within an application might not be possible (if how WLAN networks are managed is any indication).

[TBD: Illustrate how easy IPSP is when only working link-local like CoAP-over-GATT does, see also <https://gitlab.com/chrysn/coap-over-gatt/-/issues/10> (<https://gitlab.com/chrysn/coap-over-gatt/-/issues/10>).]

- * GoldenGate [goldengate]: This introduces significant network overhead, and burdens the end user device application with shipping a full network stack that is executed in a position where it can not integrate fully with the operating system's network stack.

Moreover, this places a retransmission layer on top of a partially reliable transport (GATT), duplicating effort and possibly aggravating congestion situations.

- * CoAP over UDP over SLIP over GATT UART [nefzger]: This is similar to the GoldenGate approach, but built on the GATT UART provided with Nordic Semiconductor's libraries.

This shares the network stack duplication and retransmission concerns of GoldenGate.

- * slipmux [I-D.bormann-t2trg-slipmux] over BLE GATT UART service:
This is similar to the previous item; the stack duplication concern is addressed, but retransmissions are still active atop of a service that already provides some reliability.

2. Terminology

3. Protocol description

3.1. Boundary conditions: GATT properties

[This section may be shortened in later iterations, but is kept around while the protocol is being developed to easily fix mistakes made from wrong assumptions.]

CoAP-over-GATT has different properties than UDP transported over the Internet:

- * Messages sent by one party are received by the other party in the order in which they are sent. There is no re-ordering.

(There is also a total order on all message exchanged between two peers, but that property is not useful because it's often not accessible through the Bluetooth stacks.)

- * There is limited reliability built into the protocol.

Data transmissions initiated by the data source can be unreliable ("write without response", "notify") or reliable ("write with response", "indicate").

The caveat with their reliability is that acknowledgements are sent by the BLE stack, without consulting with the application. (This is not only done for simplicity but also for power efficiency: There is only a short time window in which the data source is listening for confirmations). Thus, these confirmations can not serve to acknowledge that the a CoAP request contained in the event was read, understood and is being processed.

The reliability mechanisms are still useful, though: Both "write" and "notify"/"indicate" update the GATT characteristic's state, and while a slow application may miss data when sent in fast succession, it is reasonable to expect from the BLE stack to deliver the last data to the application when no more data is sent.

- * Reads and writes may be subtly confused: When a characteristic is written to, and it is read before the BLE server application has had time to interact with its BLE stack, the written value may be echoed back at read time.

This is likely not problematic when "notify"/"indicate" is used instead of polling reads, but it seems prudent to take precautions.

3.2. Requests and responses

CoAP-over-GATT uses a GATT Characteristics to transport request and response messages. Similar CoAP-over-UDP it offers both reliable and unreliable transfer and message deduplication, but as GATT's properties (see Section 3.1) differ from UDP's, it uses a different serialization and a different kind of message IDs.

Tokens are used like with other CoAP transports, and allow keeping multiple requests active at the same time.

A GATT server announces service of UUID 8df804b7-3300-496d-9dfa-f8fb40a236bc (abbreviated US in this document), with one or more pairs of characteristics of UUID 8bf52767-5625-43ca-a678-70883a366866 (the downstream characteristic, abbreviated UCD) and ab3720c8-7fc0-41f8-aa2a-9a45c2c01a4b (the upstream characteristic, abbreviated UCU) through BLE advertisements from a BLE peripheral (typically a constrained device), which are discovered by a BLE central (typically an end user device). The server and client roles of CoAP and GATT are independent of each other: either BLE participant can send requests in a CoAP client role.

It is expected that as this document matures, shorter (16 or 32 bit) identifiers will be requested and assigned. [See also <https://gitlab.com/chrysn/coap-over-gatt/-/issues/7> (<https://gitlab.com/chrysn/coap-over-gatt/-/issues/7>).]

3.2.1. Message sub-layer

At the UCU/UCD pair of CoAP-over-GATT characteristics, each party maintains a single bit Message ID (initialized at 1 when a connection is created), and the last Message ID sent by the peer (initialized at 0 when a connection is created).

Messages are serialized as GATT values. The GATT client sends a message by writing it to UCD (reliably using the "write with response" or unreliably using "write without response" operation); the GATT server sends them reliably using an "indicate" or unreliably "notify" event on UCU. The serialization format is the same for all, and illustrated in Figure 1:

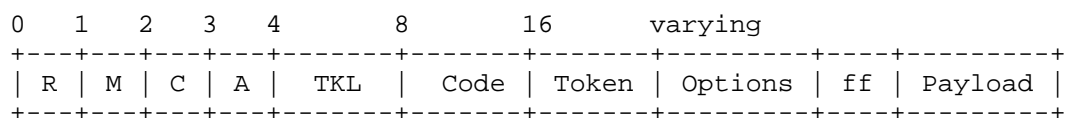


Figure 1: Components of a message

- * a single message description byte, compose of 4 bits R (reserved), M (Message ID), C (Confirm) and A (Acknowledge ID), followed by 4 bits of token length (TKL).
- * Code, token, options, payload marker and payload as in [RFC7252].

Unlike there, there is no 16-bit Message ID field (a similar role is taken by bits M and A), and in empty messages, the code is not sent.

The bits are set as follows:

- * The R bit is reserved for future extensions; it MUST be written as 0, and when a value of 1 is written, the whole written value MUST be ignored.
- * The Message ID bit is always set to the current Message ID of the sender.
- * The Confirm bit is set if the sender asks the peer to acknowledge that the message has been noted.
- * The Acknowledge ID is always set to the peer's last sent Message ID that had the Confirm bit set.

When receiving a message with the C bit set, the recipient MUST eventually send a response message with radio reliability.

3.2.2. Minimal use of the message sub-layer

Unless an implementation sends fast bursts of updates or large quantities of data and tunes for their throughput, it can use a simple subset of the message sub-layer functionality and still interoperate with any peer.

Such an application needs to

- * always send reliably (i.e., use Write with Response and Inform)
- * set the Message ID to 0 on the first message it sends,
- * set the Confirm bit on every non-empty message it sends (and leaves it unset on empty messages),
- * wait for the peer to send a message with Acknowledge matching the last Message ID it sent before sending another non-empty message, and
- * always send an acknowledgement right after receiving a non-empty message with the Confirm bit set.

This way, there is no need to keep track of whether the Confirm bit is strictly necessary, and whether more messages may be sent with the same message ID.

3.2.3. Using the message sub-layer

[This section reflects ongoing experimentation with the above serialization format and rules. Senders may use other patterns as long as they do not stall their peer by not sending any messages after the Confirm bit was set.]

To send a message unreliably in terms of CoAP transmission, a sender sets its latest Message ID in the M bit, sets C to 0, and populates the remaining bits per the rules above. It then sends the message unreliably on the radio (it may be sent reliably, especially when the peer set the C bit before). After a CoAP-unreliable message, the sender may send more CoAP-unreliable messages. It should avoid sending multiple messages in the same connection event (because the peer's BLE stack would be likely to not pass on the earlier message).

To send a message reliably in terms of CoAP transmission, a sender sets its latest Message ID in the M bit, sets C to 1, and populates the remaining bits per the rules above. It then sends the message reliably on the radio (it may send unreliably if a message is expected from the peer soon, but then needs to be prepared to send the same message again). After sending that message, the sender does not send any other message until a message is received with A equal to the sent message's M bit. The sender may need to send the very same message again if no earlier transmission of the message happened reliably. [Do we need to give timing guidance here? Probably not, because it only happens if there is some expectation in the first place.] The sender may cancel the transmission by sending an empty message with the same M and C bits, or by sending different message with these bits (which are then all unreliable transmissions).

When receiving a message with the C bit set, it is up to the recipient when to send the radio-reliable message. If it is expected that a radio-reliable message will be sent soon, it is permissible and useful to send unrelated unreliable messages that already account for the set C bit in their A bit.

3.2.4. Message deduplication

CoAP-over-GATT participants MUST ignore a message arriving at a characteristic if it is identical to the one received previously in the same connection. (The first message is never ignored).

Recipients MAY limit deduplication to the message up to and including the token. This has no practical impact on reliable transmission: If a sender desires one precise message to be sent, it will increment M, set C, and not send another message until A has matched M.

Note that it is not possible to send two identical consecutive messages unreliably. When sending identical requests, the sender may vary the token. Sending identical responses generally is rarely significant, even with the generalized [I-D.bormann-core-responses], because the mechanism to make responses "non-matching" in that document's terminology typically incurs variation. When it does not, but the repetition is still significant, sending the messages reliably becomes necessary.

3.2.5. Requests and responses

CoAP requests and responses are built on the message sub-layer as they are in [RFC7252]: requests are sent with a token chosen by the CoAP client, and the CoAP server sends a response with the same token.

Responses and message-layer acknowledgments can happen in the same message. Unlike in [RFC7252], there is no association between a request and its message ID: Any message may serve as an acknowledgement; it is always only the token that matches requests to responses.

3.2.6. Fragmentation

Attribute values are limited to 512 Bytes ([bluetooth52] Part F Section 3.2.9), practically limiting blockwise operation ([RFC7959]) to size exponents to 4 (resulting in a block size of 256 byte). Even smaller messages might enhance the transfer efficiency when they avoid fragmentation at the L2CAP level. [TBD: Verify:]

3.2.7. Multiple characteristics

If a server provides multiple UCU and UCD typed characteristics, they form pairs in the sequence in which they are listed. By using them in parallel, multiple messages can be sent without waiting for individual confirmation. This is similar to using RFC7252 with NSTART > 1, and may be used by the GATT client if the GATT server lists multiple pairs of UCU/UCD characteristics. The GATT server can send messages only through UCU characteristics on which the GATT client enabled "indicate" or "notify"; if the GATT client does not support multiple characteristics, it will just pick any pair, and only enable them on the pair's UCU.

Each characteristic has its independent message ID bits. All characteristics of a service share a single token space, and responses need not necessarily be sent on the characteristic the request was sent on.

The use of multiple characteristics is primarily practical when large amounts of data are to be transferred, or when low-latency notifications are required while simultaneously sending reliable messages. These transfers can utilize much of BLE's bandwidth because they make it easy to send much data within a single BLE connection event.

Implementers are encouraged to benchmark their applications and show that their throughput is limited by the number of characteristics used before supporting multiple characteristics.

3.2.8. Communication example

The example illustrated in Figure 2 shows an observation request with reliable and unreliable responses. It chooses the most typical configuration where the GATT server is also the BLE peripheral (and thus sends advertisements). The GATT client is also the CoAP client here.

GATT server	GATT client
-------------	-------------

```

Send BLE advertisement with one UCU and one UCD ----->

(Pairing in Just-Works mode and discovery not illustrated)

<----- Write+Resp. M=1 C=1 A=0 T="01" GET /temp, Observe: 0

(The server sends temperature values unreliably for some time)

Notify M=1 C=0 A=1 T="01" 2.05 Content, Obs: 1, "22属C" --->
Notify M=1 C=0 A=1 T="01" 2.05 Content, Obs: 2, "21属C" --->
<----- Write+Resp. M=0 C=1 A=0 T="02" GET /model

Indicate M=1 C=1 A=0 T="02" 2.05 Content, "ExampleScan" -->
<----- Write+Resp. M=0 C=0 A=1 empty

Notify M=0 C=0 A=0 T="01" 2.05 Content, Obs: 3, "20属C" --->

(At this point, the temperature isn't changing for some time,
and the server sends a reliable notification)

Indicate M=0 C=1 A=0 T="01" 2.05 Content, Obs: 4, "20属C" ->
<----- Write+Resp. M=0 C=0 A=0 empty

```

Figure 2: Example message flow

3.3. Addresses

The URI scheme associated with CoAP over GATT is "coap" as per the recommendation of Section 6 of [I-D.ietf-core-transport-indication]. The default value of Uri-Host is the MAC address of the CoAP server, in hexadecimal encoding, followed by .ble.arpa.

// The use of .ble.alt as defined in [RFC9476] was considered instead
 // of .ble.arpa, but rejected for lack of management of its
 // subdomains. Language from the .alt specification may be used when

// it comes to describing how this is not disturbing DNS operations.

User information and port are always absent with this scheme.

Assembling the URI of a request for the discovery resource of a BLE device with the MAC address 00:11:22:33:44:55 would thus be assembled, under the rules of Section 6.4 of [RFC7252], to `coap://001122334455.ble.arpa/.well-known/core`.

These addresses do not convey a particular version of CoAP-over-GATT (or, more generally, scheme of transporting CoAP over other Bluetooth mechanisms): When the referenced device is found, Bluetooth's discovery mechanisms are used, and discovering the service US indicates availability of CoAP-over-GATT as specified in this document.

Locally defined host or service name registries may be used to create names that are more suitable for human interaction. For DNS, which is widely used for this purpose, no record types are registered that map to Bluetooth MAC addresses at the time of writing.

Note that on some platforms (e.g. Web Bluetooth [webbluetooth]), the peer's or the own address may not be known application. They may come up with an application-internal registered name component (e. g. `coap://id-SomeInternalIdentifier.alt/.well-known/core`, in this case using the `.alt` zone from [RFC9476]), but must be aware that those can not be expressed towards anything outside the local stack -- the same way they would avoid using IPv6 zone identifiers or URIs whose host name is localhost.

3.3.1. Use with persistent addresses

When services are meant to provide long-lived and universally usable URIs, addresses based on MAC addresses can be impractical, because they fluctuate on hardware changes. (Moreover, privacy mechanisms on the device or the platform can render them unusable even before hardware changes).

In the absence of a usable host or service name registry, implementers may opt for non-GATT addresses right away. Section 2.4.1 of [I-D.ietf-core-transport-indication] provides the means to advertise a different canonical address, and to announce availability of that advertised service on the present transport.

When long-lived addresses circumvent privacy preserving measures, considerations concerning the tracking of devices [are TBD along the lines of "don't make it discoverable to unauthorized sources, and in case of doubt let the peer show its credentials first"].

4. Further development

4.1. Compression and reinterpretation of non-CoAP characteristics

The use of SCHC is being evaluated in combination with CoAP-over-GATT; the device can use the characteristic UUID to announce the static context used.

Together with non-traditional response forms ([I-D.bormann-core-responses] and contexts that expand, say, a numeric value 0x1234 to a message like

```
2.05 Content
Response-For: GET /temperature
Content-Format: application/senml+cbor
Payload (in JSON-ish equivalent):
[
  {1 /* unit */: "K", 2 /* value */: 0x1234}
]
```

This enables a different use case than dealing with limited environments: Accessing BLE devices via CoAP without application specific gateways. Any required information about the application can be expressed in the SCHC context.

4.2. Additional use of advertisements

In the current specification, advertisements are used to indicate that CoAP-over-GATT is being used.

Possible extensions include:

- * Additional data can be used to identify the device.

Suitable fields are Service Data (currently unavailable in the Web Bluetooth implementation) and Manufacturer Data (requires a registered Company Identifier of a SIG member; available in the Web Bluetooth implementation).

Those can allow quick selection of a device to contact even when selection by MAC address is unavailable (as are .

These identifiers can also form a suitable host component through a manufacturer specific (Manufacturer Data) or to-be-defined (Service Data) mechanism.

Service Data is currently not a practical attribute, both because it is not implemented in any web browser, and because it practically requires a 16-bit UUID as the US (otherwise it exceeds available space).

- * Some resource metadata might already be transported in advertisements.

These would need to be compact (in the order of magnitude of 10 bytes or less), and could contain data otherwise only discovered by querying the .well-known/core resource, or (hashes of) AS and audience values for ACE to facilitate connection creation with a device known by its managed identity.

- * Advertisements could contain broadcast CoAP messages.

Given that these non-traditional responses can not have embedded requests (as defined in [I-D.bormann-core-responses]) due to size constraints, a mechanism such as [I-D.ietf-core-observe-multicast-notifications] could be used to distribute some consensus request: Devices would learn that there is a consensus multicast request, and convey any response by sending BLE advertisements.

In some cases, a consensus request may be global: For example, devices willing to participate in an [I-D.ietf-lake-authz] mediated enrollment would respond to a consensus request that is an empty POST to their /.well-known/edhoc resource. A field in an advertisement "responding" to that may then contain an EDHOC message 1, and receivers can attempt to process it without further radio traffic, only establishing a GATT connection upon success. The kind of request that elicited that response, as well as possibly some constant prefix (fixed method or suites) is encoded in the type of field (possibly a dedicated service with own service data).

4.3. Protocol details

- * Is there any good reason to allow read operations?

A GATT client that is waiting for a Confirm bit to be acknowledged might attempt a Read (for the case that the confirmation arrived in an unreliable message), but might just as well perform the last write again.

Reading would be more efficient (because it can happen without application intervention, and no data is sent), but the added complexity might not be worth the enhancements.

- * Fragmentation. If the current approach of requiring devices to support large MTU sizes turns out to be impractical, or if GATT level fragmentation vastly outperforms CoAP fragmentation, it may be necessary to use composite reads and writes on GATT.

Care has to be taken to use only operations supported by [webbluetooth]: that API does not expose reads with offsets.

Offset based fragmentation may also be incompatible with the write-with-response approach suggested for reliability.

- * Usability from WebBluetooth

WebBluetooth clients may be unaware that two protocol instances are running between the client and the server at the same time, without any indication on the BLE side.

Is there anything this protocol can do to help the clients discover (or even resolve) the situation?

See also <https://gitlab.com/chrysn/coap-over-gatt/-/issues/9> (<https://gitlab.com/chrysn/coap-over-gatt/-/issues/9>).

5. IANA considerations

5.1. ble.arpa

IANA is asked to create two new reserved domain names in the .arpa name space as described in [rfc6761]: the suffixes .ble.arpa.

The expectation for Application Software are that no DNS resolution is attempted; instead, the hexadecimal prefix is processed into a 6-byte binary address, and any operation on that address is pointed to the Bluetooth Low Energy device with the indicated MAC address.

The Domain Reservation Considerations from Section 5 of [RFC6761] for both domains are:

- * Users: Users are not expected to recognize those names as special, merely as distinct from other names.
- * Application Software: Writers of application software are expected to pass them on to their CoAP implementation. CoAP implementations are expected to recognize them as Bluetooth addresses, and use their Bluetooth addresses and MUST NOT pass them on to DNS based resolvers (unless the API resolver happens to explicitly support resolution into BLE addresses, see below).

- * Name resolution APIs and libraries: Name resolution APIs and libraries MAY indicate that .ble.arpa names resolve to the BLE MAC address literals encoded inside them (but no details for this are specified in known resolution APIs or libraries). Otherwise, they SHOULD report them as NXDOMAIN.
- * Caching DNS Servers: Caching DNS servers MAY recognize the special domains and report them as NXDOMAIN. Otherwise, they will cache the .arpa DNS servers' responses.
- * Authoritative DNS Servers: Authoritative DNS servers MAY recognize the special domains and report them as NXDOMAIN. Otherwise, they will cache the .arpa DNS servers' responses.
- * DNS Server Operators: No impact on DNS server operators is expected.
- * DNS Registries/Registrars: Any changes to .ble.arpa or .ble-sd.arpa go through updates to this document and IANA.

6. Security considerations

All data received over GATT is considered untrusted; secure communication can be achieved using OSCORE [RFC8613].

Physical proximity can not be inferred from this means of communication.

7. References

7.1. Normative References

- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<https://www.rfc-editor.org/rfc/rfc7252>>.
- [RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <<https://www.rfc-editor.org/rfc/rfc7595>>.
- [rfc6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <<https://www.rfc-editor.org/rfc/rfc6761>>.

7.2. Informative References

- [RFC7668] Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., and C. Gomez, "IPv6 over BLUETOOTH(R) Low Energy", RFC 7668, DOI 10.17487/RFC7668, October 2015, <<https://www.rfc-editor.org/rfc/rfc7668>>.
- [webbluetooth]
Grant, R. and O. Ruiz-Henrriquez, "Web Bluetooth", 24 February 2020, <<https://webbluetoothcg.github.io/web-bluetooth/>>.
- [goldengate]
Fitbit, Inc, "Golden Gate", 2020, <<https://fitbit.github.io/golden-gate/>>.
- [nefzger] Matthias Nefzger, "Talk CoAP to me IoT over Bluetooth Low Energy", 1 March 2021, <<https://www.maibornwolff.de/en/blog/talk-coap-me-iot-over-bluetooth-low-energy>>.
- [RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, DOI 10.17487/RFC8323, February 2018, <<https://www.rfc-editor.org/rfc/rfc8323>>.
- [RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <<https://www.rfc-editor.org/rfc/rfc8613>>.
- [RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <<https://www.rfc-editor.org/rfc/rfc7959>>.
- [bluetooth52]
"Bluetooth Core Specification v5.2", 31 December 2019, <https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=478726>.
- [I-D.bormann-t2trg-slipmux]
Bormann, C. and T. Kaupat, "Slipmux: Using an UART interface for diagnostics, configuration, and packet transfer", Work in Progress, Internet-Draft, draft-bormann-t2trg-slipmux-03, 4 November 2019, <<https://datatracker.ietf.org/doc/html/draft-bormann-t2trg-slipmux-03>>.

[I-D.bormann-core-responses]

Bormann, C. and C. Amsss, "CoAP: Non-traditional response forms", Work in Progress, Internet-Draft, draft-bormann-core-responses-06, 20 October 2025, <<https://datatracker.ietf.org/doc/html/draft-bormann-core-responses-06>>.

[I-D.ietf-core-transport-indication]

Amsss, C. and M. S. Lenders, "CoAP Transport Indication", Work in Progress, Internet-Draft, draft-ietf-core-transport-indication-09, 7 July 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-core-transport-indication-09>>.

[RFC9476] Kumari, W. and P. Hoffman, "The .alt Special-Use Top-Level Domain", RFC 9476, DOI 10.17487/RFC9476, September 2023, <<https://www.rfc-editor.org/rfc/rfc9476>>.

[I-D.ietf-core-observe-multicast-notifications]

Tiloca, M., Hglund, R., Amsss, C., and F. Palombini, "Observe Notifications as CoAP Multicast Responses", Work in Progress, Internet-Draft, draft-ietf-core-observe-multicast-notifications-13, 20 October 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-core-observe-multicast-notifications-13>>.

[I-D.ietf-lake-authz]

Selander, G., Mattsson, J. P., Vuini, M., Fedrecheski, G., and M. Richardson, "Lightweight Authorization using Ephemeral Diffie-Hellman Over COSE (ELA)", Work in Progress, Internet-Draft, draft-ietf-lake-authz-07, 2 March 2026, <<https://datatracker.ietf.org/doc/html/draft-ietf-lake-authz-07>>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <<https://www.rfc-editor.org/rfc/rfc6761>>.

Appendix A. Change log

Since draft-amsuess-core-coap-over-gatt-08:

- * Unmodified re-upload as adopted document.

Since -07:

- * Removed ble-sd.arpa and demoted other Service Data use.

Service Data is currently not available in Web Bluetooth implementations.

- * Limited message deduplication to the header.

This makes the protocol implementable on constrained devices without a doubled buffer.

- * Small additions around multiple characteristics and discovery interaction with addressing.

Since -06:

- * Sketch usage example with EDHOC message 1 in beacons.
- * Restructured to group "further development" points together.
- * Discourage multi-characteristic operation unless necessary.
- * Minor clarifications.

Since -05:

- * Use `coap://${MAC}.ble.arpa` instead of `coap+gatt://`.
- * Apply template to IANA considerations.

Since -04:

- * Point out `.arpa` / `.alt` considerations.

Since -03:

- * Define semantics of service data field, define `ble-sd.arpa` for that purpose.
- * Switch to `.arpa` names for MAC addresses for consistency with service data names.
- * Use one characteristic per data direction. This
 - simplifies implementations on platforms with little control over change events,
 - removes the necessity to process the R bit, and
 - frees up that bit in messages.

- * Add communication example.
- * Reference more open issues, including intention to get shorter IDs.

Since -02:

- * Message format extended by a leading byte, the option to have a token. This enables role reversal and concurrent requests.
- * The UC identifier was changed to reflect the incompatible change in protocol.
- * A section on used BLE properties was added.
- * A section providing outlook on other data for advertisements was added.

Since -01:

- * Point out (possibly conflicting) development directions.
- * Describe URI scheme more completely, including persistent addresses.
- * Aim for standards track.
- * Describe rejected alternative approaches.

Since -00:

- * Add note on SCHC possibilities.

Author's Address

Christian Amsss
Austria
Email: christian@amsuess.com