Workgroup: CoRE
Internet-Draft:
draft-amsuess-core-coap-over-gatt-02
Published: 12 July 2022
Intended Status: Standards Track
Expires: 13 January 2023
Authors: C. Amsüss

CoAP over GATT (Bluetooth Low Energy Generic Attributes)

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.

Note to Readers

Discussion of this document takes place on the CORE 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://gitlab.com/chrysn/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 13 January 2023.

Copyright Notice

Table of Contents

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: 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).

* 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 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 reliability.

2. Terminology

3. Protocol description

3.1. Requests and responses

[This section is not thought through or implemented yet, and could probably end up very different.]

CoAP-over-GATT uses individual GATT Characteristics to model a reliable request-response mechanism. Therefore, it has no message types or message IDs (in which it resembles CoAP-over-TCP [RFC8323]), and no tokens. In the place of tokens, different Bluetooth characteristics (comparable to open ports in IP based

networks) can be used. All messages use GATT to ensure reliable transmission.

A GATT server announces service of UUID 8df804b7-3300-496d-9dfa-f8fb40a236bc (abbreviated US in this document), with one or more characteristics of UUID 2a58fc3f-3c62-4ecc-8167-d66d4d9410c2 (abbreviated UC).

[Right now, this only supports requests from the GATT client to the GATT server; role reversal might be added later.]

A client can start a CoAP request by writing to the UC characteristic a sequence composed of a single code byte, any options encoded in the option format of [RFC7252] Section 3.1, optionally followed by a payload marker and the request payload.

After the successful write, the client can read the response back from the server on the same characteristic. The client may need to attempt reading the characteristic several times until the response is ready, and may subscribe to indications to get notifiied when the response is ready.

The server does needs to keep the response readable after it has been read, for the server can not know whether the read was completed by the client.

If the request and initial response establish an observation, the client may keep reading; the server may keep the latest notification available indefinitely (especially if it turns out that "has been read successfully" is hard to determine) or make it readable only once for each new state.

Once the client writes a new request to a UC characteristic, any later reads pertain to that request, and any observation previously established is cancelled implicitly.

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.

If a server provides multiple OC typed characteristics, parallel requests or observations are possible; otherwise, this transport is limited to a single pending request.

3.1.1. Development directions

Three major concerns may need addressing in future iterations of this protocol:

* Role reversal.

  This may be implemented by adding a GATT server to the central, or by multiplexing requests and responses onto a single read and write channel.

* Response reliability.

  When multiple responses are sent to a request (e.g. when using [I-D.tiloca-core-groupcomm-proxy], or more generally [I-D.bormann-core-responses]) of which all need to be delivered, or if role reversal is implemented by multiplexing, the GATT server needs to know when a message has been read; the GATT mechanisms do not provide that information.

  Previously, this was not deemed relevant, as for the original non-traditional responses, observation notifications [RFC7641], only eventual consistency is relevant.

  One option is to replace reads with write-with-response operations, and to introduce a flag that marks previously read messages as received. This is essentially building a 1-bit message ID mechanism. (No longer IDs are necessary, because messages on GATT are not reordered on the network).

* 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.

* Concurrent requests. If a multiplexing approach is chosen for role reversal, the current setup of multiple characteristics for multiple requests may become obsolete.

  A possible solution is to re-introduce tokens, in a message format similar to that of CoAP-over-WebSockets [RFC8323].

3.2. Addresses

The URI scheme associated with CoAP over GATT is "coap+gatt". The default value of Uri-Host is the MAC address of the CoAP server, in hexadecimal encoding, with the dash character ("-") separating the bytes. [Some bikeshedding is expected on these details.]

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+gatt://00-11-22-33-44-55/.well-known/core.

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+gatt://id-SomeInternalIdentifier/.well-known/core), 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.

3.2.1. Scheme-free alternative

As an alternative to the abovementioned scheme, a zone in .arpa could be registered to use addresses like

coap://001122334455.ble.arpa/.well-known/core

where the .ble.arpa address do not resolve to any IP addresses.

[Accepting this will require a .arpa registering IANA consideration to replace the URI one.]

3.2.2. 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. [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, CoAP-over-GATT. If the device is not generally reachable, the canonical address might also be unreachable (see [I-D.ietf-core-transport-indication] section "Unreachable canonical origin address").

When long-lived addresses circumvent privacy preserving measures, considerations concering 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"].

3.3. 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. IANA considerations

4.1. Uniform Resource Identifier (URI) Schemes

IANA is asked to enter a new scheme into the "Uniform Resource Identifier (URI) Schemes" registry set up in [RFC7595]:

*URI Scheme: "coap+gatt"

*Description: CoAP over Bluetooth GATT (sharing the footnote of coap+tcp)

*Well-Known URI Support: yes, analogous to [RFC7252]

5. 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.

6. References

6.1. 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/info/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/info/rfc7595>.

6.2. 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/info/rfc7668>.

[webbluetooth] Grant, R. and O. Ruiz-Henríquez, "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/info/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/info/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/info/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://www.ietf.org/archive/id/draft-bormann-t2trg-slipmux-03.txt>.

[I-D.tiloca-core-groupcomm-proxy] Tiloca, M. and E. Dijk, "Proxy Operations for CoAP Group Communication", Work in Progress, Internet-Draft, draft-tiloca-core-groupcomm-proxy-06, 7 March 2022, <https://www.ietf.org/archive/id/draft-tiloca-core-groupcomm-proxy-06.txt>.

[I-D.bormann-core-responses] Bormann, C. and C. Amsüss, "CoAP: Non-traditional response forms", Work in Progress, Internet-Draft, draft-bormann-core-responses-01, 3 February 2022, <https://www.ietf.org/archive/id/draft-bormann-core-responses-01.txt>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/info/rfc7641>.

[I-D.ietf-core-transport-indication]
Amsüss, C., "CoAP Protocol Indication", Work in Progress, Internet-Draft, draft-ietf-core-transport-indication-01, 11 July 2022, <https://www.ietf.org/archive/id/draft-ietf-core-transport-indication-01.txt>.

Appendix A. Change log

Since -01:

* Point out (possibly conflicting) development directions.

* Describe URI scheme more completely, including persistent addresses.

* Aim for standards track.

* Describe rejeced alternative approaches.

Since -00:

*Add note on SCHC possibilities.

Author's Address

Christian Amsüss
Austria

Email: christian@amsuess.com