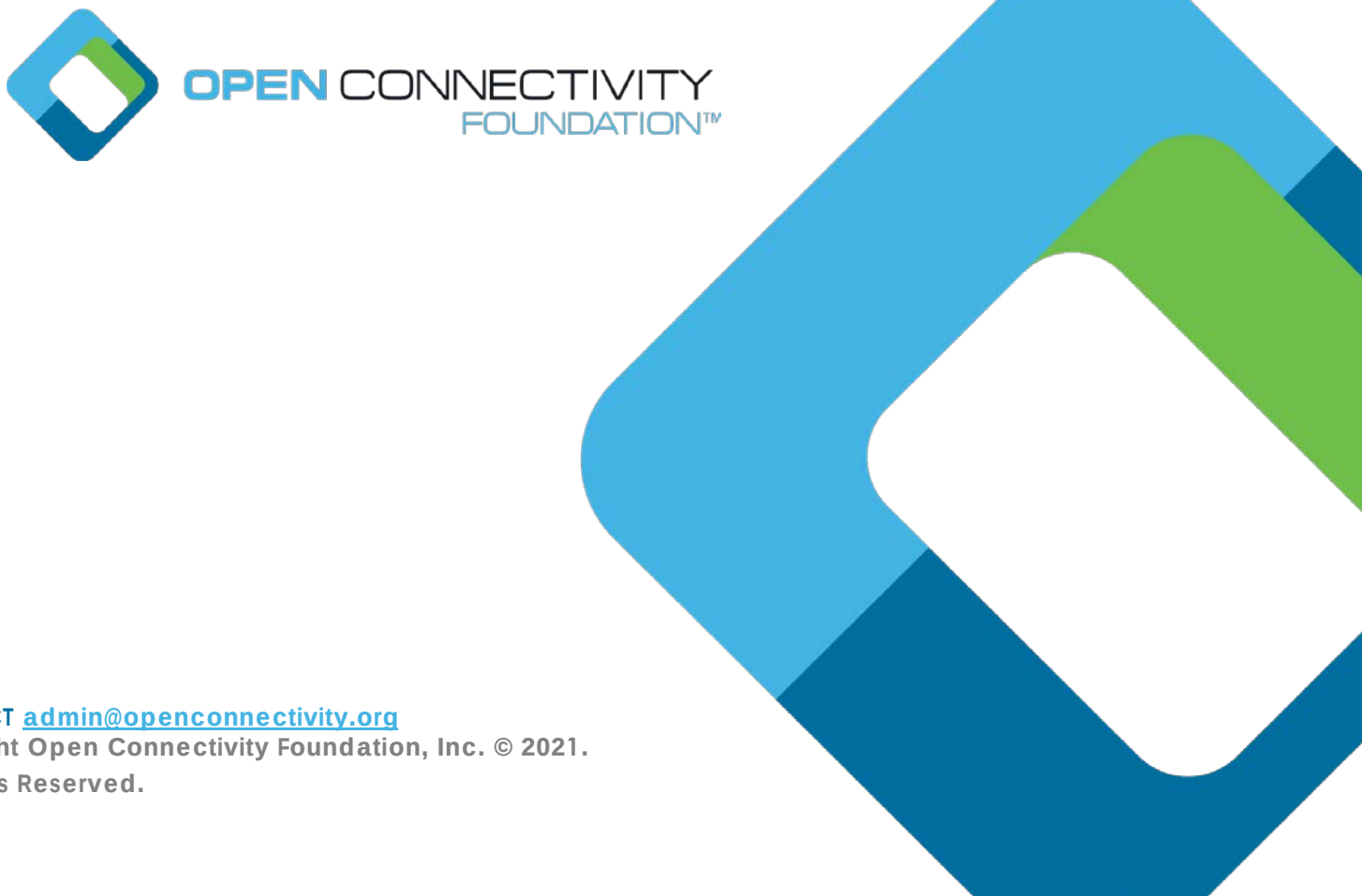
OCF Resource to BLE Mapping Specification

VERSION 2.2.3 | April 2021



CONTACT admin@openconnectivity.org

Legal Disclaimer

NOTHING CONTAINED IN THIS DOCUMENT SHALL BE DEEMED AS GRANTING YOU ANY KIND OF LICENSE IN ITS CONTENT, EITHER EXPRESSLY OR IMPLIEDLY, OR TO ANY INTELLECTUAL PROPERTY OWNED OR CONTROLLED BY ANY OF THE AUTHORS OR DEVELOPERS OF THIS DOCUMENT. THE INFORMATION CONTAINED HEREIN IS PROVIDED ON AN "AS IS" BASIS, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE AUTHORS AND DEVELOPERS OF THIS SPECIFICATION HEREBY DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, STATUTORY OR AT COMMON LAW, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. OPEN CONNECTIVITY FOUNDATION, INC. FURTHER DISCLAIMS ANY AND ALL WARRANTIES OF NON-INFRINGEMENT, ACCURACY OR LACK OF VIRUSES.

The OCF logo is a trademark of Open Connectivity Foundation, Inc. in the United States or other countries. *Other names and brands may be claimed as the property of others.

CONTENTS

Figures

Tables

Introduction

This document, and all the other parts associated with this document, were developed in response to worldwide demand for smart home focused Internet of Things (IoT) devices, such as appliances, door locks, security cameras, sensors, and actuators; these to be modelled and securely controlled, locally and remotely, over an IP network.

While some inter-device communication existed, no universal language had been developed for the IoT. Device makers instead had to choose between disparate frameworks, limiting their market share, or developing across multiple ecosystems, increasing their costs. The burden then falls on end users to determine whether the products they want are compatible with the ecosystem they bought into, or find ways to integrate their devices into their network, and try to solve interoperability issues on their own.

In addition to the smart home, IoT deployments in commercial environments are hampered by a lack of security. This issue can be avoided by having a secure IoT communication framework, which this standard solves.

The goal of these documents is then to connect the next 25 billion devices for the IoT, providing secure and reliable device discovery and connectivity across multiple OSs and platforms. There are multiple proposals and forums driving different approaches, but no single solution addresses the majority of key requirements. This document and the associated parts enable industry consolidation around a common, secure, interoperable approach.

1 Scope

This document provides detailed mapping information between BLE (Bluetooth Low Energy) and OCF defined Resources.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Adopted Bluetooth Profiles, Services, Protocols and Transports
https://www.bluetooth.com/specifications/adopted-specifications

Bluetooth Core Specification 4.0
https://www.bluetooth.com/specifications/bluetooth-core-specification

ISO/IEC 30118-1 Information technology -- Open Connectivity Foundation (OCF) Specification -- Part 1: Core specification
https://www.iso.org/standard/53238.html
Latest version available at: https://openconnectivity.org/specs/OCF_Core_Specification.pdf

ISO/IEC 30118-2 Information technology – Open Connectivity Foundation (OCF) Specification – Part 2: Security specification
https://www.iso.org/standard/74239.html
Latest version available at: https://openconnectivity.org/specs/OCF_Security_Specification.pdf

ISO/IEC 30118-3 Information technology – Open Connectivity Foundation (OCF) Specification – Part 3: Bridging specification
https://www.iso.org/standard/74240.html
Latest version available at: https://openconnectivity.org/specs/OCF_Bridging_Specification.pdf

ISO/IEC 30118-4 Information technology – Open Connectivity Foundation (OCF) Specification – Part 4: Resource Type specification
https://www.iso.org/standard/74241.html
Latest version available at:
https://openconnectivity.org/specs/OCF_Resource_Type_Specification.pdf

ISO/IEC 30118-5 Information technology – Open Connectivity Foundation (OCF) Specification – Part 5: Device specification
https://www.iso.org/standard/79389.html
Latest version available at: https://openconnectivity.org/specs/OCF_Device_Specification.pdf

Derived Models for Interoperability between IoT Ecosystems, Stevens & Merriam, March 2016
https://www.iab.org/wp-content/IAB-uploads/2016/03/OCF-Derived-Models-for-Interoperability-Between-IoT-Ecosystems_v2-examples.pdf

IETF RFC 4122, *A Universally Unique IDentifier (UUID) URN Namespace*, July 2005
https://www.rfc-editor.org/info/rfc4122

3 Terms, definitions, symbols and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 30118-1, ISO/IEC 30118-2, and ISO/IEC 30118-3 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

– ISO Online browsing platform: available at https://www.iso.org/obp

– IEC Electropedia: available at http://www.electropedia.org/

3.1.1
GATT-based Profile
BLE profile using procedures and operating models provided by GATT profile

3.2 Symbols and abbreviated terms

ATT	Attribute protocol
GAP	Generic Access Profile
GATT	Generic Attribute profile

4 Document conventions and organization

4.1 Conventions

In this document a number of terms, conditions, mechanisms, sequences, parameters, events, states, or similar terms are printed with the first letter of each word in uppercase and the rest lowercase (e.g., Network Architecture). Any lowercase uses of these words have the normal technical English meaning.

In this document, to be consistent with the IETF usages for RESTful operations, the RESTful operation words CRUDN, CREATE, RETRIVE, UPDATE, DELETE, and NOTIFY will have all letters capitalized. Any lowercase uses of these words have the normal technical English meaning.

4.2 Notation

In this document, features are described as required, recommended, allowed or DEPRECATED as follows:

Required (or shall or mandatory).

These basic features shall be implemented to comply with the Mapping Specification. The phrases "shall not", and "PROHIBITED" indicate behavior that is prohibited, i.e. that if performed means the implementation is not in compliance.

Recommended (or should).

These features add functionality supported by the Mapping Specification and should be implemented. Recommended features take advantage of the capabilities the Mapping Specification, usually without imposing major increase of complexity. Notice that for compliance testing, if a recommended feature is implemented, it shall meet the specified requirements to be in compliance with these guidelines. Some recommended features could become requirements in the future. The phrase "should not" indicates behavior that is permitted but not recommended.

Allowed (or allowed).

These features are neither required nor recommended by the Mapping Specification, but if the feature is implemented, it shall meet the specified requirements to be in compliance with these guidelines.

Conditionally allowed (CA)

The definition or behaviour depends on a condition. If the specified condition is met, then the definition or behaviour is allowed, otherwise it is not allowed.

Conditionally required (CR)

The definition or behaviour depends on a condition. If the specified condition is met, then the definition or behaviour is required. Otherwise the definition or behaviour is allowed as default unless specifically defined as not allowed.

DEPRECATED

Although these features are still described in this document, they should not be implemented except for backward compatibility. The occurrence of a deprecated feature during operation of an implementation compliant with the current document has no effect on the implementation's operation and does not produce any error conditions. Backward compatibility may require that a feature is implemented and functions as specified but it shall never be used by implementations compliant with this document.

Strings that are to be taken literally are enclosed in "double quotes".

Words that are emphasized are printed in *italic*.

5 Theory of Operation

5.1 Interworking Approach

The interworking between the BLE defined services/characteristics model and OCF defined Resources is modelled using the derived model syntax described in Derived Models for Interoperability between IoT Ecosystems.

5.2 Mapping Syntax

Within the defined syntax for derived modelling used by this document there are two blocks that define the actual Property-Property equivalence or mapping. These blocks are identified by the keywords "x-to-ocf" and "x-from-ocf". Derived Models for Interoperability between IoT Ecosystems does not define a rigid syntax for these blocks; they are free form string arrays that contain pseudo-coded mapping logic.

In this document, Python (version >= 3.0) syntax is used to describe translation rules.

The JSON skeleton shows typical translation block used in the derived models.

```
"<BLE Service Name>" : {
        "type": "object",
        "properties": {
                "<a value field in BLE Characteristic value>" : {
                        "x-ocf-conversion" : {
                                "x-ocf-alias": "<corresponding OCF Resource type>",
                                "x-to-ocf": [
                                        ...
                                        ...
                                ],
                                "x-from-ocf": [
                                        "N/A"
                                ]
                        }
                }
        }
}
```

– <BLE Service Name>: this is fully qualified name of a BLE Service (e.g. "org.bluetooth.characteristic.blood_pressure_measurement")

– <a value field in BLE Characteristic value>: a Characteristic value is byte stream which is composed of multiple value fields. “A value field in BLE Characteristic value” is a description for one of them.

– <corresponding OCF Resource type>: an OCF Resource type which is corresponding to this BLE Service.

– “N/A”: in BLE Bridging, most of the BLE devices are read only. So there is no specific value to be written to the BLE devices from OCF Devices. Therefore, nothing is described in “x-from-ocf” translation clause. “N/A” is used to describe this case.

6 BLE Translation

6.1 Operational Scenarios

6.1.1 Introduction

The overall goal is to make Bridged BLE GATT Servers appear to OCF Clients as if they were native OCF Servers in the local network or cloud environment.

“Deep translation” between specific BLE Profile and OCF Device is specified in clause 9. Figure 1 shows an overview of the BLE Bridge Platform and its general topology. The BLE Bridging Function supports Asymmetric bridging. It exposes BLE GATT Servers to OCF Clients. Each Bridged BLE GATT Server shall be represented as a Virtual OCF Server.

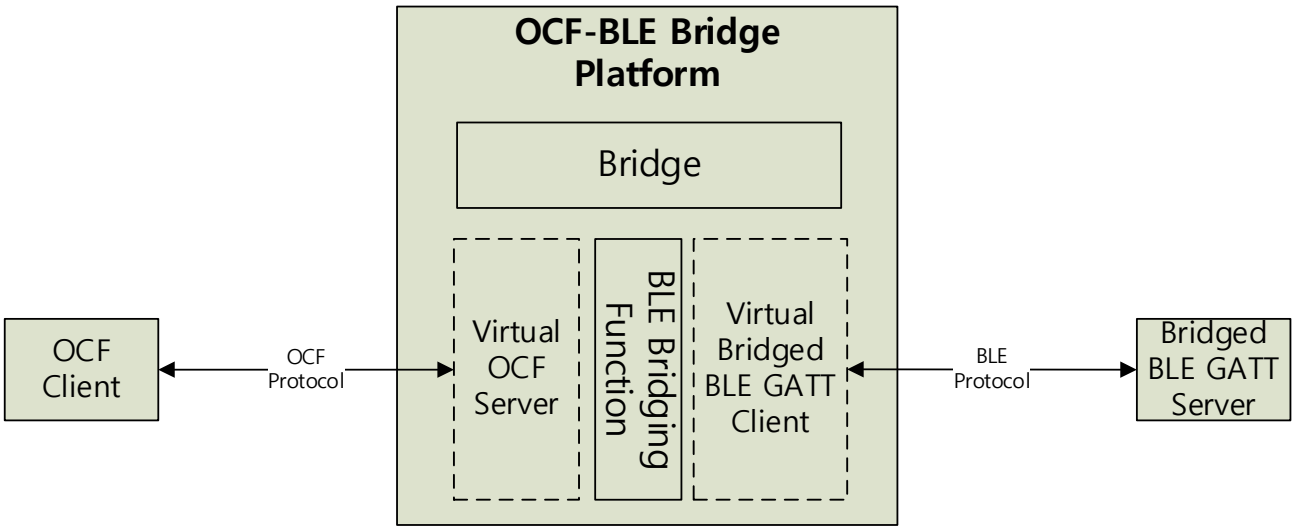


Figure 1 – OCF-BLE Bridge Platform Components

6.1.2 Use case for BLE Bridging

Figure 2 shows a use case for an OCF Client and BLE GATT Server. An OCF Client on a smartphone reads a BLE thermometer device through an OCF-BLE Bridge Platform. Any connectivity that OCF supports is used for communications between the OCF Client and the OCF-BLE Bridge Platform. The OCF Client can communicate with OCF-BLE Bridge Platform through OCF Cloud.

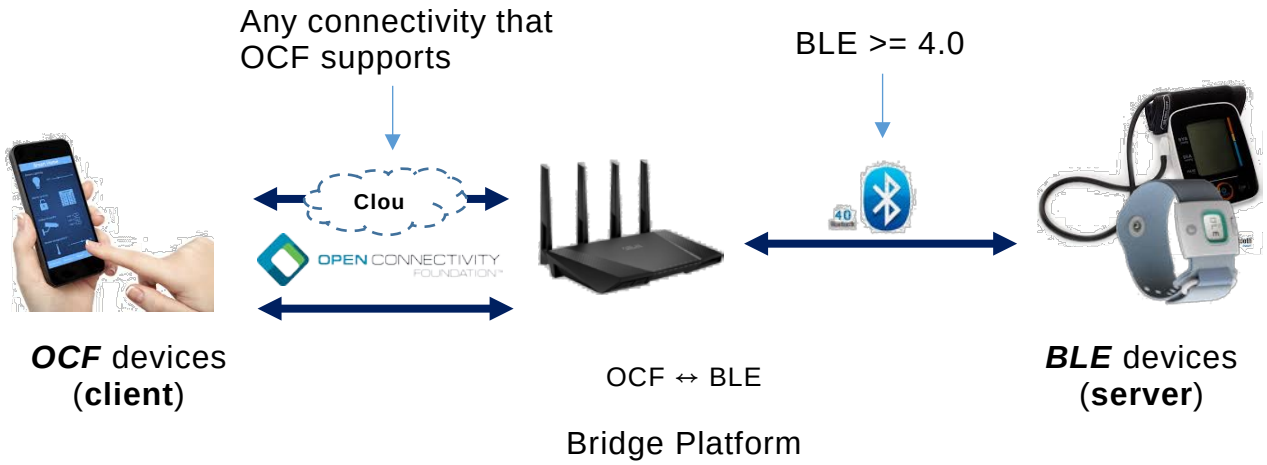


Figure 2 – BLE Bridging use case in real life

6.2 Requirements specific to BLE Bridging Function

6.2.1 General

OCF-BLE Bridge Platform shall satisfy clause 5.2 General Requirements of ISO/IEC 30118-3.

A BLE Bridging Function supports asymmetric bridging. It exposes BLE GATT server to OCF Clients only. Therefore, it shall play a BLE GATT client role. (This is a requirement so that users can expect that a certified OCF Bridge Platform will be able to talk to any BLE GATT server device, without the user having to buy some other device.).

6.2.2 Requirements specific to BLE

The version of Bluetooth SIG core specification that this document refers to is 4.0 or higher (see Bluetooth Core Specification 4.0). Bluetooth BR/EDR is not included in the scope of this document.

6.2.3 Exposing BLE GATT Servers to OCF Clients

6.2.3.1 General

The requirements in this clause apply when using algorithmic translation, and by default apply to deep translation unless the relevant requirements for such deep translation specifies otherwise.

Basic translation rule between BLE Service/Characteristic model and OCF Resource model is described in Table 1.

Table 1 – Translation rule between BLE and OCF data model

From BLE	mapping count	To OCF	mapping count
GATT-based profile	n	OCF Device	1
Service	1	OCF Resource	n
Characteristic	1	OCF Resource Property	n
Characteristic Descriptor	1	OCF Notification on/off option	1

One or more BLE GATT-based profiles should be mapped to one Virtual OCF Server (e.g. Health Thermometer profile (HTP) is mapped to Body Thermometer Device ("oic.d.body.thermometer")). A BLE Service should be mapped to one or more OCF Resources (e.g. Health Thermometer Service is mapped to Temperature ("oic.r.body.temperature") and Body Location for temperature ("oic.r.body.location.temperature")). Each Characteristic of BLE Service should be mapped to one or more Properties of OCF Resource (if there is no BLE Characteristic corresponding to an OCF Property, default value should be used). Table 2 is a translation example of this rule. Figure 3 provides an illustration of this rule.

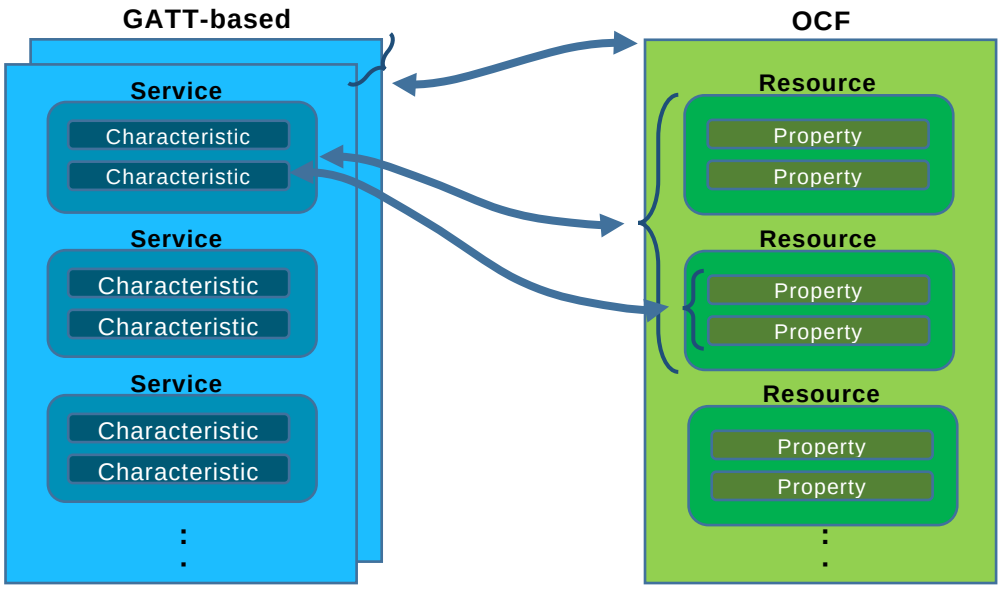


Figure 3 – Translation mapping rule illustration

Table 2 – BLE to OCF translation example (Blood Pressure Device)

	BLE	OCF
BLE Profile → OCF Device	Blood Pressure Profile (BLP)	Blood Pressure Monitor Device ("oic.d.bloodpressuremonitor")
BLE Service → OCF Resource	Blood Pressure Measurement Service ("org.bluetooth.service.blood_pressure")	Blood Pressure ("oic.r.blood.pressure")
		Pulse Rate ("oic.r.pulserate")
	Device Information Service ("org.bluetooth.service.device_information")	Device ("oic.wk.d") Platform ("oic.wk.p")
BLE Characteristic → OCF Resource Property	Blood Pressure Measurement ("org.bluetooth.characteristic.blood_pressure_measurement")	"oic.r.blood.pressure.systolic" "oic.r.blood.pressure.diastolic" "oic.r.blood.pressure.map" "oic.r.blood.pressure.units"
		"oic.r.pulserate.pulserate"

Figure 4 shows an example for 1:N mapping between BLE Characteristic and OCF Properties. In this case, multiple fields in “Blood Pressure Measurement Service” are mapped into the Properties of OCF Resources ("oic.r.pulserate", "oic.r.blood.pressure").

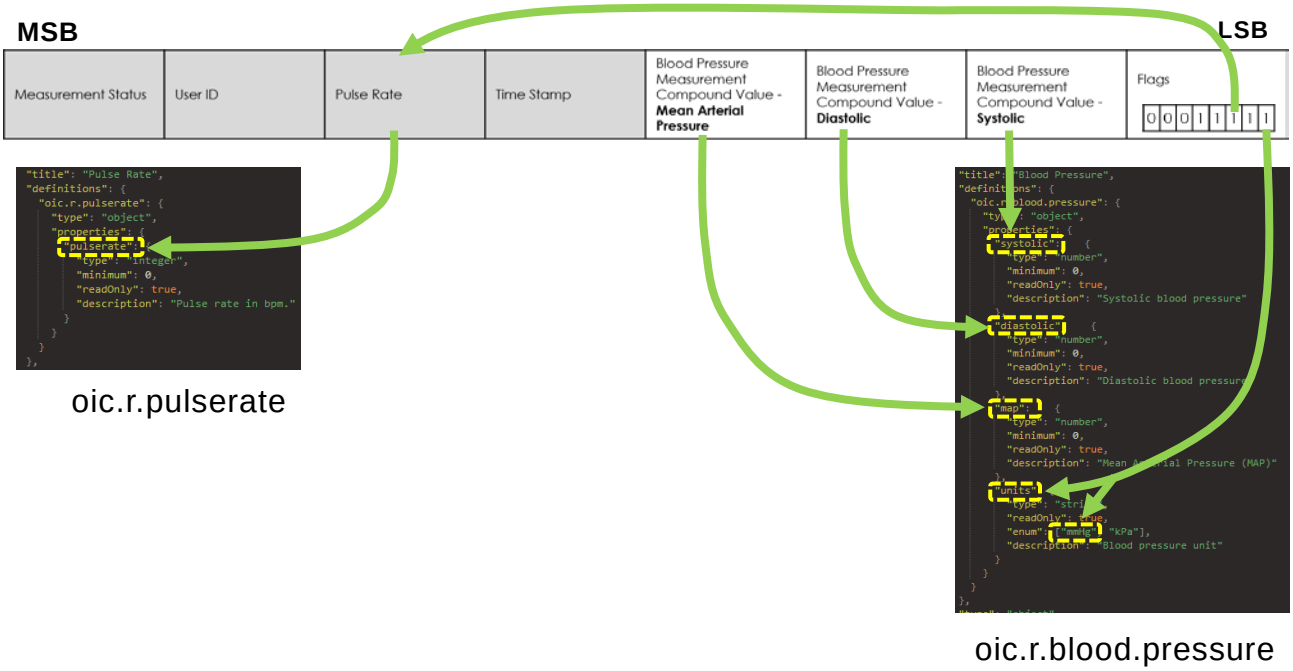


Figure 4 – An example for 1:N mapping between BLE Characteristic and OCF Properties

6.2.3.2 Translation for well-defined set

6.2.3.2.1 General

If a BLE Profile is in a well-defined set, translation should be done as follows. Table 3 is the list of BLE GATT-based Profiles which have corresponding OCF Resources as of now.

Table 3 – BLE GATT-based Profile – OCF Resource mapping

BLE GATT-based Profile	BLE Service	OCF Resource		OCF Device Type
		Atomic Measurement Resource Type	Resource Type	
Blood Pressure Profile	Blood Pressure Service	"oic.r.bloodpressuremonitor-am"	"oic.r.blood.pressure"	"oic.d.bloodpressuremonitor"
			"oic.r.pulserate"	
	Device Information Service		"oic.wk.d"	
			"oic.wk.p"	
Glucose Profile	Glucose Service	"oic.r.glucosemeter-am"	"oic.r.glucose"	"oic.d.glucosemeter"
			"oic.r.glucose.carb"	
			"oic.r.glucose.exercise"	
			"oic.r.glucose.hba1c"	
			"oic.r.glucose.health"	
			"oic.r.glucose.meal"	
			"oic.r.glucose.medication"	
			"oic.r.glucose.samplelocation"	

			"oic.r.glucose.tester"	
	Device Information Service		"oic.wk.d"	
			"oic.wk.p"	
Health Thermometer Profile	Health Thermometer Service	"oic.r.bodythermometer-am"	"oic.r.temperature"	"oic.d.bodythermometer"
			"oic.r.body.location.temperature"	
	Device Information Service		"oic.wk.d"	
			"oic.wk.p"	
Weight Scale Profile	Weight Scale Service	"oic.r.bodyscale-am"	"oic.r.weight"	"oic.d.bodyscale"
			"oic.r.bmi"	
			"oic.r.height"	
			"oic.r.body.fat"	
			"oic.r.body.water"	
			"oic.r.body.slm"	
			"oic.r.body.ffm"	
	Device Information Service		"oic.wk.d"	
			"oic.wk.p"	

6.2.3.2.2 URI for Virtual OCF Resource

This clause describes how the URI for a Virtual OCF Resource is derived.

Case 1: a BLE Service is mapped to an OCF Resource:

– */<BLE Service name without prefix "org.bluetooth.service">*, (e.g. BLE Service "Fitness Machine (org.bluetooth.service.fitness_machine)": /fitness_machine)

Case 2: a BLE Service is mapped to multiple OCF Resources. If corresponding multiple OCF Resources are grouped by Collection (or Atomic Measurement Collection), URI should be as follows:

– URI for Collection Resource: */<BLE Service name without prefix "org.bluetooth.service">* (e.g. BLE Service "Health Thermometer (org.bluetooth. service.health_thermometer)": /health_thermometer)

– URI for each OCF Resource link: */<OCF Resource Type value of corresponding linked Resource without prefix "oic.r">* (e.g. /temperature for "oic.r.temperature", /body.location.temperature for "oic.r.body.location.temperature")

If corresponding multiple OCF Resources are not grouped by Collection, URI should be as follows:

– URI for each OCF Resource: /<BLE Service name without prefix "org.bluetooth.service">/<OCF Resource Type value of corresponding Resource without prefix "oic.r">

Table 4 provides an example applying the rules defined in this clause.

Table 4 – URI mapping example

	BLE	OCF
URI	Health Thermometer Service ("org.bluetooth.service.health_thermometer")	/health_thermometer (for Atomic Collection Resource) /temperature (for "oic.r.temperature") /body.location.temperature (for "oic.r.body.location.temperature")

6.2.3.2.3 Common Properties of Resource Type

Resource Type (“rt”, Mandatory): value of “rt” in corresponding OCF Resource specified in ISO/IEC 30118-4.

Interface (“if”, Mandatory): value of “if” in corresponding OCF Resource specified in ISO/IEC 30118-4.

6.2.3.2.4 Platform Resource (“rt” of "oic.wk.p")

Platform ID (“pi”, Mandatory): since BLE device does not provide a mandatory unique “name” (or id) which can be used to generate name-based UUID described in IETF RFC 4122 clause 4.3, randomly-generated UUID described in IETF RFC 4122 clause 4.4 should be used for Platform ID.

Manufacturer Name (“mnmn”, Mandatory): if Device Information Service is implemented “manufacturer_name_string” Characteristic should be used, or “<device_name> by unknown” should be used as default value (<device_name> is a Characteristic of GAP).

6.2.3.2.5 Device Resource (“rt” of "oic.wk.d")

Spec Version (“icv”, Mandatory): Spec version of ISO/IEC 30118-1 that the Bridging Function implements should be used.

Device UUID (“di”, Mandatory): as specified in ISO/IEC 30118-2, the value of the “di” Property of OCF Devices (including Virtual OCF Devices) shall be established as part of On-boarding of that Virtual OCF Device.

Data Model Version (“dmv”, Mandatory): spec version of the vertical specification this device data model is implemented to should be used. Syntax is “<vertical>.major.minor”.

Protocol Independent ID (“piid”, Mandatory): randomly-generated UUID described in IETF RFC 4122 clause 4.4 should be used for "piid".

6.2.3.3 Exposing a BLE GATT Server as a Virtual OCF Server

Table 5 shows how OCF Device Properties as specified in ISO/IEC 30118-1, should be derived, typically from fields specified in BLE Device Information Service (Spec Type: "org.bluetooth.service.device_information", Service ID: 0x180A) and Generic Access Service (Spec Type: "org.bluetooth.service.generic_access", Service ID: 0x1800).

Table 5 – "oic.wk.d" Resource Type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory?	From BLE Device Service Characteristic value	BLE Description	BLE Mandatory?
(Device) Name	"n"	Human friendly name For example, “Bob’s Thermostat”	Y	Device Name (Generic Access)		Y
Spec Version	"icv"	Spec version of ISO/IEC 30118-1 this Device is implemented to, The syntax is "core.major.minor”]	Y	(none)	Bridging Function should return its own value	

Device UUID	"di"	Unique identifier for Device. This value shall be as defined in ISO/IEC 30118-2 for Device UUID.	Y	(none)	Use as defined in ISO/IEC 30118-2	
Protocol-Independent ID	"piid"	Unique identifier for OCF Device (UUID). Randomly-generated UUID described in IETF RFC 4122 clause 4.4 should be used for piid	Y	(none)	(none)	
Data Model Version	"dmv"	Spec version(s) of the vertical specifications this Device data model is implemented to. The syntax is a comma separated list of "<vertical>.major.minor"]. <vertical> is the name of the vertical (e.g. sh for Smart Home)	Y	(none)	(none)	
Localized Descriptions	"ld"	Detailed description of the Device, in one or more languages. This Property is an array of objects where each object has a "language" field and a "value" field containing the Device description in the indicated language.	N	(none)	(none)	
Software Version	"sv"	Version of the Device software.	N	Software Revision String (Device Information)	This characteristic represents the software revision for the software within the Device.	N
Manufacturer Name	"dmn"	Name of manufacturer of the Device, in one or more languages. This Property is an array of objects where each object has a "language" field and a "value" field containing the manufacturer name in the indicated language.	N	Manufacturer Name String (Device Information)	This characteristic represents the name of the manufacturer of the Device.	N
Model Number	"dmno"	Model number as designated by manufacturer.	N	Model Number String (Device Information)	This characteristic represents the model number that is assigned by the Device vendor.	N

Regarding configuration resource ("oic.wk.con"), it is not created on the Virtual OCF Server since that information/interaction is not supported on BLE side.

Table 6 shows how platform Properties, as specified in ISO/IEC 30118-1, are derived, typically from fields specified in BLE Device Information Service and Generic Access Service.

Table 6 – "oic.wk.p" Resource Type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory?	From BLE Device Service Characteristic value	BLE Description	BLE Mandatory?
Platform ID	"pi"	Unique identifier for the physical platform (UIUID); this shall be a UUID in accordance with IETF RFC 4122. It is recommended that the UUID be created using the random generation scheme (version 4 UUID) specific in the RFC.	Y	(none)	(none)	
Manufacturer Name	"mnmn"	Name of manufacturer (not to exceed 16 characters)	Y	Manufacturer Name String (Device Information). if Device Information Service is not implemented “<device_name> by unknown” should be used as default value (<device_name> is a Characteristic of GAP)	This characteristic represents the name of the manufacturer of the Device.	N
Manufacturer Details Link (URL)	"mnml"	URL to manufacturer (not to exceed 32 characters)	N	(none)	(none)	
Model Number	"mnmo"	Model number as designated by manufacturer	N	Model Number String (Device Information)	This characteristic represents the model number that is assigned by the Device vendor.	N
Date of Manufacture	"mndt"	Manufacturing date of Device	N	(none)	(none)	
Platform Version	"mnpv"	Version of platform – string (defined by manufacturer)	N	Software Revision String (Device Information)	This characteristic represents the software revision for the software within the Device.	N

OS Version	"mnos"	Version of platform resident OS – string (defined by manufacturer)	N	(none)	BLE device usually has no OS.	
Hardware Version	"mnhw"	Version of platform hardware	N	Hardware Revision String (Device Information)	This characteristic represents the hardware revision for the hardware within the Device.	N
Firmware version	"mnfv"	Version of Device firmware	N	Firmware Revision String (Device Information)	This characteristic represents the firmware revision for the firmware within the Device.	N
Support URL	"mnsl"	URL that points to support information from manufacturer	N	(none)	(none)	
System Time	"st"	Reference time for the Device	N	(none)	(none)	
Vendor ID	"vid"	Vendor defined string for the platform. The string is freeform and up to the vendor on what text to populate it.	N	Manufacturer Name String (Device Information)	This characteristic represents the name of the manufacturer of the Device.	N

Table 7 shows how configurable OCF Platform Properties, as specified in Table 16 in ISO/IEC 30118-1, should be derived as follows, if a BLE device does not implement Device Information Service, "oic.wk.con.p" should not be created on the Virtual OCF Server.

Table 7 – "oic.wk.con.p" Resource Type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory?	From BLE Device Service Characteristic value	BLE Description	BLE Mandatory?
Platform Names	"mnpn"	Platform Identifier	N	Manufacturer Name String (Device Information)	This characteristic represents the name of the manufacturer of the Device.	

No BLE Service equivalence exist for factory reset or restart, so there is no Characteristics for "oic.wk.mnt" Properties "Factory_Reset" and "Reboot", so mapping for "oic.wk.mnt" is omitted.

6.2.3.4 On-the-fly Translation

If a BLE Profile is not in Table 3 (not belong to a well-defined set), a BLE Bridging Function does not translate it (on-the-fly translation is not supported).

6.2.3.5 Protocol translation between BLE and OCF

Adopted Bluetooth Profiles, Services, Protocols and Transports describes not only Service/Characteristic data model but also Features how to manipulate it. GATT Features define how GATT-based data exchanges takes place. The GATT features are used when we translate OCF CRUDN into BLE protocol and vice versa.

Table 8 shows translation rule between BLE GATT Feature and OCF CRUDN. When a BLE Bridging Function receives CREATE/DELETE request from OCF Client, it shall return corresponding error (i.e. 4.xx or 5.xx) because there are no corresponding Features for them. If a BLE Bridging Function receives RETRIEVE/UPDATE request from OCF Client, it shall translate it into Characteristic Value Read/Characteristic Value Write respectively. NOTIFY request from OCF Client shall be translated into Characteristic Descriptor Value Write, and Characteristic Value Notification/Indication from BLE GATT Server shall be translated into NOTIFICATION response.

Table 8 – Protocol translation rule between BLE and OCF

BLE GATT Feature	OCF CRUDN
N/A (Not Supported)	CREATE
Characteristic Value Read	RETRIEVE
Characteristic Value Write	UPDATE
N/A (Not Supported)	DELETE
Characteristic Descriptor Value Write	NOTIFY request
Characteristic Value Notification/Indication	NOTIFICATION response

6.2.3.6 Illustrative OCF to BLE translation flows

6.2.3.6.1 Initialization

Figure 5shows the initial pairing procedure.

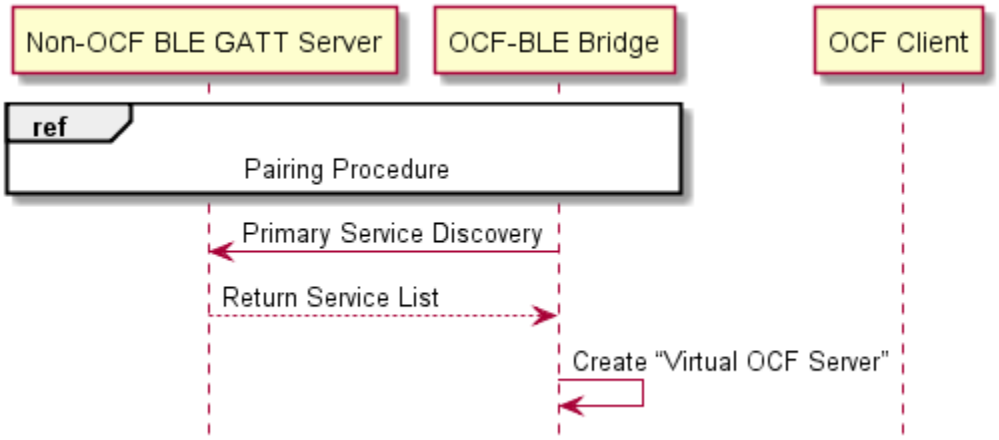


Figure 5 – Initialization

6.2.3.6.2 Resource Discovery

Figure 6 shows the resource discovery procedure.

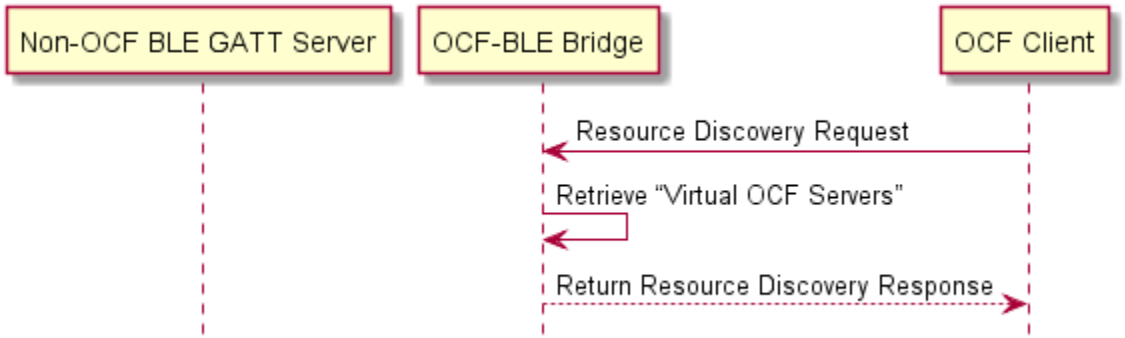


Figure 6 – Resource Discovery

6.2.3.6.3 Create Resource

Figure 7 illustrates Resource creation.

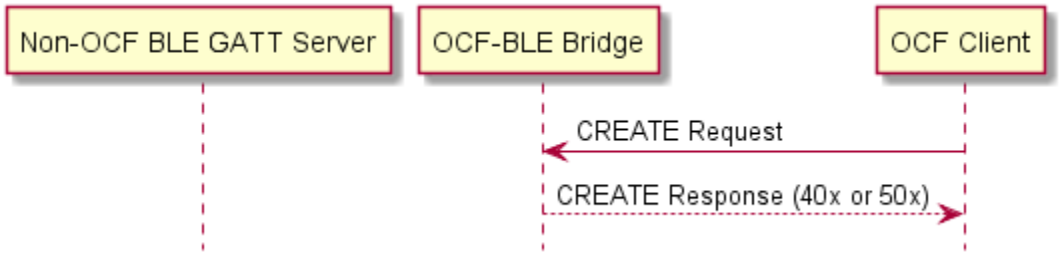


Figure 7 – Create Resource

6.2.3.6.4 Retrieve Resource

Figure 8 illustrates Resource RETRIEVAL.

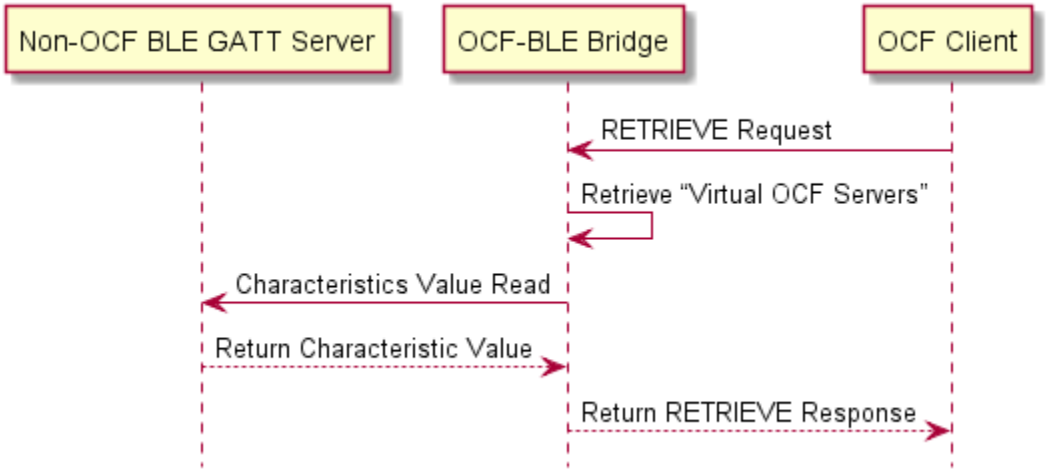


Figure 8 – Retrieve Resource

6.2.3.6.5 Update Resource

Figure 9 illustrates Resource UPDATE.

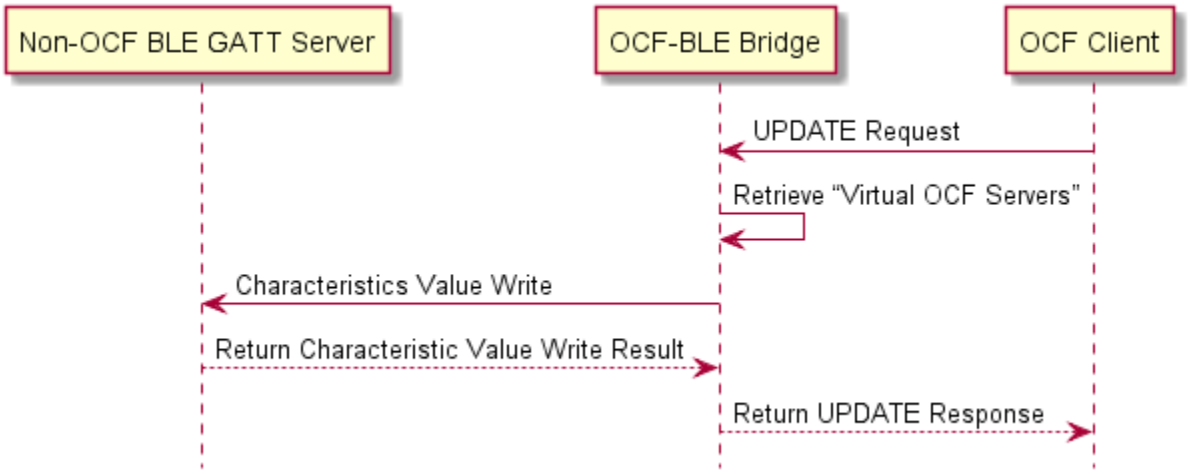


Figure 9 – Update Resource

6.2.3.6.6 Delete Resource

Figure 10 illustrates Resource DELETE. Note that this only applies to Resources that were created.

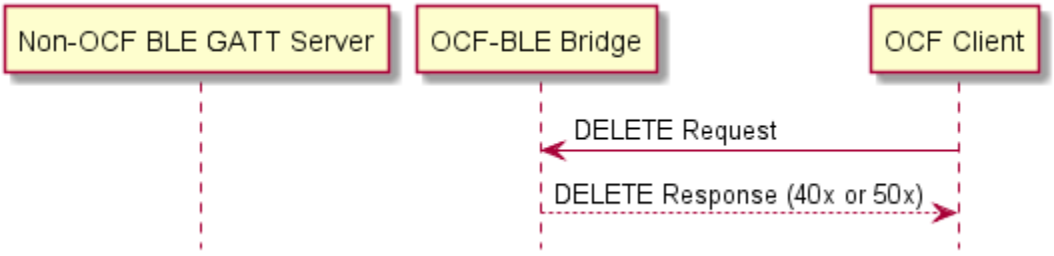


Figure 10 – Delete Resource

6.2.3.6.7 Set Notification & Send Notification

Figure 11 illustrates the establishment and sending of a notification.

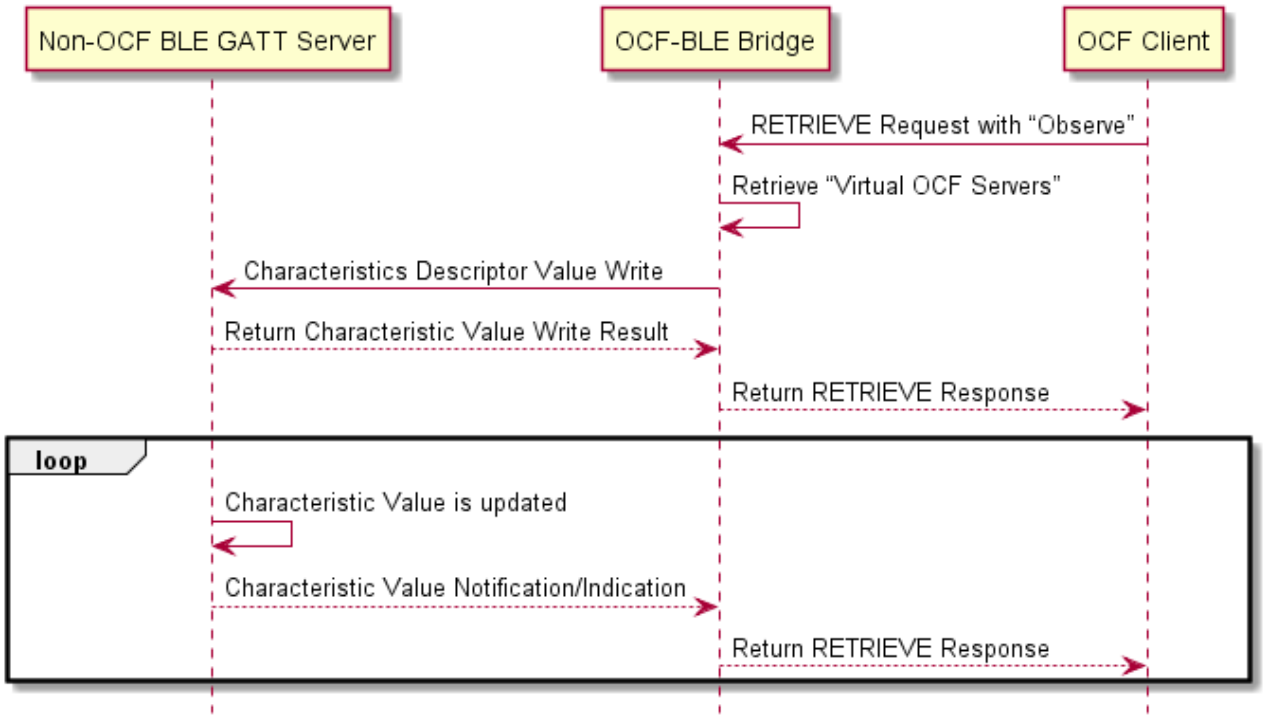


Figure 11 – Set Notification and send Notification

6.2.3.7 Error handling

If a BLE operation fails, the Bridging Function sends an appropriate OCF error response to the OCF Client. it constructs an appropriate OCF error message (e.g., diagnostic payload if using CoAP) from the BLE error name and error code (if any), using the form "<error name>: <error message>", with the <error name> taken from the ATT error code field and the <error message> taken from the ATT error name, and the error code for the OCF network set to an appropriate value.

7 Device Type Mapping

7.1 Introduction

This clause contains the mappings to OCF Device Types.

7.2 BLE Profile to OCF Device Types

Table 9 captures the equivalency mapping between BLE Profile and OCF defined Device Types. The minimum Resource sets for each OCF Device is provided in ISO/IEC 30118-5.

Table 9 – BLE Profile to OCF Device Type Mapping

BLE GATT-based Profile	OCF Device Type
Blood Pressure Profile	"oic.d.bloodpressuremonitor"
Glucose Profile	"oic.d.glucosemeter"

Health Thermometer Profile	"oic.d.bodythermometer"
Weight Scale Profile	"oic.d.bodyscale"

8 BLE Profile to Resource Equivalence

8.1 Introduction

This clause lists the complete set of applicable BLE Profiles and provides the equivalent OCF Resource Type(s) to which the BLE Profiles map.

8.2 BLE Services to OCF Resources

Table 10 captures the equivalency mapping between BLE Services and OCF defined Resource Types (see ISO/IEC 30118-4). Detailed Property by Property mappings are provided in clause 9.

Table 10 – BLE Services to OCF Resource Type Mapping

BLE Service	OCF Resource	
	Atomic Measurement Resource Type	**Resource Type**
Blood Pressure Service	"oic.r. bloodpressuremonitor-am"	"oic.r.blood.pressure"
		"oic.r.pulserate"
Device Information Service		"oic.wk.d"
		"oic.wk.p"
Glucose Service	"oic.r.glucosemeter-am"	"oic.r.glucose"
		"oic.r.glucose.carb"
		"oic.r.glucose.exercise"
		"oic.r.glucose.hba1c"
		"oic.r.glucose.health"
		"oic.r.glucose.meal"
		"oic.r.glucose.medication"
		"oic.r.glucose.samplelocation"
		"oic.r.glucose.tester"
Device Information Service		"oic.wk.d"
		"oic.wk.p"
Health Thermometer Service	"oic.r.bodythermometer-am"	"oic.r.temperature"
		"oic.r.body.location.temperature"
Device Information Service		"oic.wk.d"
		"oic.wk.p"
Weight Scale Service	"oic.r.bodyscale-am"	"oic.r.weight"
		"oic.r.bmi"
		"oic.r.height"
		"oic.r.body.fat"

		"oic.r.body.water"
		"oic.r.body.slm"
		"oic.r.body.ffm"
Device Information Service		"oic.wk.d"
		"oic.wk.p"

9 Detailed Mappings

9.1 Introduction

This clause provides an API and mapping description that aligns with the Derived Modelling syntax described in Derived Models for Interoperability between IoT Ecosystems for all services/characteristics and Resources that are within scope.

9.2 Blood Pressure Mapping

9.2.1 Derived model

The derived model: "org.bluetooth.characteristic.blood_pressure_measurement".

9.2.2 Property definition

Table 11 provides the detailed per Property mapping for "org.bluetooth.characteristic.blood_pressure_measurement".

Table 11 – The Property mapping for "org.bluetooth.characteristic.blood_pressure_measurement".

BLE Property name	OCF Resource	To OCF	From OCF
blood_pressure_measurement[length - 3 : length - 1]	oic.r.blood.pressure	def ieee11073_Sfloat_2_Float(sfloat_value):# reserved value for Infinity or NaN (Not a Number)reserved_float_values = {0x07FE:math.inf, # +INFINITY0x07FF:math.nan, # NaN (Not a Number)0x0800:math.nan, # NRes (Not at this Resolution)0x0801:math.nan, # Reserved for future0x0802:-math.inf # -INFINITY}mantissa = sfloat_value & 0x0FFFexponent = sfloat_value >> 12if (exponent >= 0x0008):exponent = -((0x000F + 1) - exponent)output = 0if (mantissa >= 0x07FE and mantissa <= 0x0802):output = reserved_float_values[mantissa]else:if (mantissa >= 0x0800):mantissa = -((0x0FFF + 1) - mantissa)magnitude = pow(10.0, exponent)output = (mantissa * magnitude)return outputlength = len(blood_pressure_measurement)flags = blood_pressure_measurement[length - 1]oic.r.blood.pressure.systolic = ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 3 : length - 1])oic.r.blood.pressure.units = "mmHg" if (flags & 0x01) else "kPa"	N/A
blood_pressure_measurement[length - 5 : length - 3]	oic.r.blood.pressure	def ieee11073_Sfloat_2_Float(sfloat_value):# reserved value for Infinity or NaN (Not a Number)reserved_float_values = {0x07FE:math.inf, # +INFINITY0x07FF:math.nan, # NaN (Not a Number)0x0800:math.nan, # NRes (Not at this Resolution)0x0801:math.nan, # Reserved for future0x0802:-math.inf # -INFINITY}mantissa = sfloat_value & 0x0FFFexponent = sfloat_value >> 12if	N/A

		(exponent >= 0x0008):exponent = -((0x000F + 1) - exponent)output = 0if (mantissa >= 0x07FE and mantissa <= 0x0802):output = reserved_float_values[mantissa]else:if (mantissa >= 0x0800):mantissa = -((0x0FFF + 1) - mantissa)magnitude = pow(10.0, exponent)output = (mantissa * magnitude)return outputlength = len(blood_pressure_measurement)oic.r.blood.pressure.diastolic = ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 5 : length - 3])	
blood_pressure_measurement[length - 7 : length - 5]	oic.r.blood.pressure	def ieee11073_Sfloat_2_Float(sfloat_value):# reserved value for Infinity or NaN (Not a Number)reserved_float_values = {0x07FE:math.inf, # +INFINITY0x07FF:math.nan, # NaN (Not a Number)0x0800:math.nan, # NRes (Not at this Resolution)0x0801:math.nan, # Reserved for future0x0802:-math.inf # -INFINITY}mantissa = sfloat_value & 0x0FFFexponent = sfloat_value >> 12if (exponent >= 0x0008):exponent = -((0x000F + 1) - exponent)output = 0if (mantissa >= 0x07FE and mantissa <= 0x0802):output = reserved_float_values[mantissa]else:if (mantissa >= 0x0800):mantissa = -((0x0FFF + 1) - mantissa)magnitude = pow(10.0, exponent)output = (mantissa * magnitude)return outputlength = len(blood_pressure_measurement)oic.r.blood.pressure.map = ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 7 : length - 5])	N/A
blood_pressure_measurement[length - 7 - timestamp_len - 2 : length - 7 - timestamp_len]	oic.r.pulserate	def ieee11073_Sfloat_2_Float(sfloat_value):# reserved value for Infinity or NaN (Not a Number)reserved_float_values = {0x07FE:math.inf, # +INFINITY0x07FF:math.nan, # NaN (Not a Number)0x0800:math.nan, # NRes (Not at this Resolution)0x0801:math.nan, # Reserved for future0x0802:-math.inf # -INFINITY}mantissa = sfloat_value & 0x0FFFexponent = sfloat_value >> 12if (exponent >= 0x0008):exponent = -((0x000F + 1) - exponent)output = 0if (mantissa >= 0x07FE and mantissa <= 0x0802):output = reserved_float_values[mantissa]else:if (mantissa >= 0x0800):mantissa = -((0x0FFF + 1) - mantissa)magnitude = pow(10.0, exponent)output = (mantissa * magnitude)return outputlength = len(blood_pressure_measurement)flags = blood_pressure_measurement[length - 1]timestamp_len = 7 if (flags & 0x02) else 0oic.r.pulserate.pulserate = ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 7 - timestamp_len - 2 : length - 7 - timestamp_len])	N/A

Table 12 provides the details of the Properties that are part of "org.bluetooth.characteristic.blood_pressure_measurement".

Table 12 – The Properties of "org.bluetooth.characteristic.blood_pressure_measurement".

BLE Property name	Type	Required	Description
blood_pressure_measurement[length - 3 : length - 1]		yes	Blood Pressure Measurement Compound Value - Systolic
blood_pressure_measurement[length - 5 : length - 3]		yes	Blood Pressure Measurement

			Compound Value - Diastolic
blood_pressure_measurement[length - 7 : length - 5]		no	Blood Pressure Measurement Compound Value - Mean Arterial Pressure
blood_pressure_measurement[length - 7 - timestamp_len - 2 : length - 7 - timestamp_len]		no	Pulse Rate

9.2.3 Derived model definition

```
{
        "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.BLP.json#",
        "$schema": "http://json-schema.org/draft-04/schema#",
        "description" : "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
reserved.",
        "title": "Blood Pressure",
        "definitions": {
                "byte": {
                        "type": "integer",
            "minimum": 0,
            "maximum": 255
                },
                "byteArray": {
                        "type": "array",
            "items": { "$ref": "#/definitions/byte" },
            "minItems": 1,
            "uniqueItems": false
                },
                "org.bluetooth.characteristic.blood_pressure_measurement" : {
                        "type" : "object",
                        "properties": {
                                "blood_pressure_measurement[length - 3 : length - 1]": {
                                        "$ref": "#/definitions/byteArray",
                                        "description": "Blood Pressure Measurement Compound Value -
Systolic",
                                        "x-ocf-conversion": {
                                                "x-ocf-alias": "oic.r.blood.pressure",
                                                "x-to-ocf": [
                                                        "def
ieee11073_Sfloat_2_Float(sfloat_value):",
                                                            "# reserved value for Infinity or NaN
(Not a Number)",
                                                            "reserved_float_values = {",
                                                                "0x07FE:math.inf, # +INFINITY",
                                                                "0x07FF:math.nan, # NaN (Not a
Number)",
                                                                "0x0800:math.nan, # NRes (Not at this
Resolution)",
                                                                "0x0801:math.nan, # Reserved for
future",
                                                                "0x0802:-math.inf # -INFINITY",
                                                            "}",
                                                            "mantissa = sfloat_value & 0x0FFF",
                                                            "exponent = sfloat_value >> 12",
                                                            "if (exponent >= 0x0008):",
                                                                "exponent = -((0x000F + 1) -
exponent)",
                                                            "output = 0",
                                                            "if (mantissa >= 0x07FE and mantissa <=
0x0802):",
                                                                "output =
reserved_float_values[mantissa]",
                                                            "else:",
                                                                "if (mantissa >= 0x0800):",
                                                                    "mantissa = -((0x0FFF + 1) -
mantissa)",
                                                                "magnitude = pow(10.0, exponent)",
```

```
                                                        "output = (mantissa * magnitude)",
                                                    "return output",
                                                "length = len(blood_pressure_measurement)",
                                                "flags = blood_pressure_measurement[length -
1]",
                                                "oic.r.blood.pressure.systolic =
ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 3 : length - 1])",
                                                "oic.r.blood.pressure.units = \"mmHg\" if
(flags & 0x01) else \"kPa\""
                                        ],
                                        "x-from-ocf": [
                                                "N/A"
                                        ]
                                }
                        },
                        "blood_pressure_measurement[length - 5 : length - 3]": {
                                "$ref": "#/definitions/byteArray",
                                "description": "Blood Pressure Measurement Compound Value -
Diastolic",
                                "x-ocf-conversion": {
                                        "x-ocf-alias": "oic.r.blood.pressure",
                                        "x-to-ocf": [
                                                "def
ieee11073_Sfloat_2_Float(sfloat_value):",
                                                    "# reserved value for Infinity or NaN
(Not a Number)",
                                                    "reserved_float_values = {",
                                                        "0x07FE:math.inf, # +INFINITY",
                                                        "0x07FF:math.nan, # NaN (Not a
Number)",
                                                        "0x0800:math.nan, # NRes (Not at this
Resolution)",
                                                        "0x0801:math.nan, # Reserved for
future",
                                                        "0x0802:-math.inf # -INFINITY",
                                                    "}",
                                                    "mantissa = sfloat_value & 0x0FFF",
                                                    "exponent = sfloat_value >> 12",
                                                    "if (exponent >= 0x0008):",
                                                        "exponent = -((0x000F + 1) -
exponent)",
                                                    "output = 0",
                                                    "if (mantissa >= 0x07FE and mantissa <=
0x0802):",
                                                        "output =
reserved_float_values[mantissa]",
                                                    "else:",
                                                        "if (mantissa >= 0x0800):",
                                                            "mantissa = -((0x0FFF + 1) -
mantissa)",
                                                        "magnitude = pow(10.0, exponent)",
                                                        "output = (mantissa * magnitude)",
                                                    "return output",
                                                "length = len(blood_pressure_measurement)",
                                                "oic.r.blood.pressure.diastolic =
ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 5 : length - 3])"
                                        ],
                                        "x-from-ocf": [
                                                "N/A"
                                        ]
                                }
                        },
                        "blood_pressure_measurement[length - 7 : length - 5]": {
                                "$ref": "#/definitions/byteArray",
                                "description": "Blood Pressure Measurement Compound Value -
Mean Arterial Pressure",
                                "x-ocf-conversion": {
                                        "x-ocf-alias": "oic.r.blood.pressure",
                                        "x-to-ocf": [
                                                "def
ieee11073_Sfloat_2_Float(sfloat_value):",
```



```
                                                    "# reserved value for Infinity or NaN
(Not a Number)",
                                                    "reserved_float_values = {",
                                                        "0x07FE:math.inf, # +INFINITY",
                                                        "0x07FF:math.nan, # NaN (Not a
Number)",
                                                        "0x0800:math.nan, # NRes (Not at this
Resolution)",
                                                        "0x0801:math.nan, # Reserved for
future",
                                                        "0x0802:-math.inf # -INFINITY",
                                                    "}",
                                                    "mantissa = sfloat_value & 0x0FFF",
                                                    "exponent = sfloat_value >> 12",
                                                    "if (exponent >= 0x0008):",
                                                        "exponent = -((0x000F + 1) -
exponent)",
                                                    "output = 0",
                                                    "if (mantissa >= 0x07FE and mantissa <=
0x0802):",
                                                        "output =
reserved_float_values[mantissa]",
                                                    "else:",
                                                        "if (mantissa >= 0x0800):",
                                                            "mantissa = -((0x0FFF + 1) -
mantissa)",
                                                        "magnitude = pow(10.0, exponent)",
                                                        "output = (mantissa * magnitude)",
                                                    "return output",
                                                "length = len(blood_pressure_measurement)",
                                                "oic.r.blood.pressure.map =
ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 7 : length - 5])"
                                            ],
                                            "x-from-ocf": [
                                                    "N/A"
                                            ]
                                    }
                                },
                                "blood_pressure_measurement[length - 7 - timestamp_len - 2 : length
- 7 - timestamp_len]": {
                                        "$ref": "#/definitions/byteArray",
                                        "description": "Pulse Rate",
                                        "x-ocf-conversion": {
                                                "x-ocf-alias": "oic.r.pulserate",
                                                "x-to-ocf": [
                                                        "def
ieee11073_Sfloat_2_Float(sfloat_value):",
                                                            "# reserved value for Infinity or NaN
(Not a Number)",
                                                            "reserved_float_values = {",
                                                                "0x07FE:math.inf, # +INFINITY",
                                                                "0x07FF:math.nan, # NaN (Not a
Number)",
                                                                "0x0800:math.nan, # NRes (Not at this
Resolution)",
                                                                "0x0801:math.nan, # Reserved for
future",
                                                                "0x0802:-math.inf # -INFINITY",
                                                            "}",
                                                            "mantissa = sfloat_value & 0x0FFF",
                                                            "exponent = sfloat_value >> 12",
                                                            "if (exponent >= 0x0008):",
                                                                "exponent = -((0x000F + 1) -
exponent)",
                                                            "output = 0",
                                                            "if (mantissa >= 0x07FE and mantissa <=
0x0802):",
                                                                "output =
reserved_float_values[mantissa]",
                                                            "else:",
                                                                "if (mantissa >= 0x0800):",
```



```
                                                                "mantissa = -((0x0FFF + 1) -
mantissa)",
                                                            "magnitude = pow(10.0, exponent)",
                                                            "output = (mantissa * magnitude)",
                                                        "return output",
                                                    "length = len(blood_pressure_measurement)",
                                                    "flags = blood_pressure_measurement[length -
1]",
                                                    "timestamp_len = 7 if (flags & 0x02) else 0",
                                                    "oic.r.pulserate.pulserate =
ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 7 - timestamp_len - 2 : length - 7 -
timestamp_len])"
                                                ],
                                                "x-from-ocf": [
                                                        "N/A"
                                                ]
                                        }
                                }
                        }
                },
        },

        "type": "object",

        "allOf": [
                { "$ref": "#/definitions/byte" },
                { "$ref": "#/definitions/byteArray" },
        { "$ref": "#/definitions/org.bluetooth.characteristic.blood_pressure_measurement" }
        ],

        "required": [
                "blood_pressure_measurement[length - 3 : length - 1]",
                "blood_pressure_measurement[length - 5 : length - 3]"
        ]
}
```

9.3 Glucose Measurement Mapping

9.3.1 Derived model

The derived model: "org.bluetooth.characteristic.glucose_measurement".

The derived model: "org.bluetooth.characteristic.glucose_measurement_context".

9.3.2 Property definition

Table 13 provides the detailed per Property mapping for "org.bluetooth.characteristic.glucose_measurement".

Table 13 – The Property mapping for "org.bluetooth.characteristic.glucose_measurement".

BLE Property name	OCF Resource	To OCF	From OCF
glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len - 10]	oic.r.glucose	def ieee11073_Sfloat_2_Float(sfloat_value):# reserved value for Infinity or NaN (Not a Number)reserved_float_values = {0x07FE:math.inf, # +INFINITY0x07FF:math.nan, # NaN (Not a Number)0x0800:math.nan, # NRes (Not at this Resolution)0x0801:math.nan, # Reserved for future0x0802:-math.inf # -INFINITY}mantissa = sfloat_value & 0x0FFFexponent = sfloat_value >> 12if (exponent >= 0x0008):exponent = -((0x000F + 1) - exponent)output = 0if (mantissa >= 0x07FE and mantissa <= 0x0802):output = reserved_float_values[mantissa]else:if (mantissa >= 0x0800):mantissa = -((0x0FFF + 1) -	N/A

		mantissa)magnitude = pow(10.0, exponent)output = (mantissa * magnitude)return outputlength = len(glucose_measurement)flags = glucose_measurement[length - 1]timeoffset_len = 2 if (flags & 0x01) else 0if (flags & 0x02) == True: glucose = ieee11073_Sfloat_2_Float(glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len - 10]) oic.r.glucose.glucose = (glucose * 1000) if (flags & 0x04) else (glucose * 0.1 * 1000 * 1000) oic.r.glucose.units = "mmol/L" if (flags & 0x04) else "mg/dL"if (flags & 0x02) == False: oic.r.glucose.glucose = 0 oic.r.glucose.units = "mmol/L"	
glucose_measurement[length - 1 - 2 - timeoffset_len - 10]	oic.r.glucose.samplelocation	length = len(glucose_measurement)flags = glucose_measurement[length - 1]timeoffset_len = 2 if (flags & 0x01) else 0if (flags & 0x02): samplelocation = { 1:"finger", 2:"ast", 3:"earlobe", 4:"ctrlsolution" } oic.r.glucose.samplelocation.samplelocation = samplelocation[glucose_measurement[length - 1 - 2 - timeoffset_len - 10] & 0xf0]	N/A

Table 14 provides the details of the Properties that are part of "org.bluetooth.characteristic.glucose_measurement".

Table 14 – The Properties of "org.bluetooth.characteristic.glucose_measurement".

BLE Property name	Type	Required	Description
glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len - 10]		yes	Glucose Concentration
glucose_measurement[length - 1 - 2 - timeoffset_len - 10]		no	Sample Location

Table 15 provides the detailed per Property mapping for "org.bluetooth.characteristic.glucose_measurement_context".

Table 15 – The Property mapping for "org.bluetooth.characteristic.glucose_measurement_context".

BLE Property name	OCF Resource	To OCF	From OCF
glucose_measurement_context[length - carb_len - extflags_len - 3 : length - 1 - extflags_len - 3]	oic.r.glucose.carb	def ieee11073_Sfloat_2_Float(sfloat_value):# reserved value for Infinity or NaN (Not a Number)reserved_float_values = {0x07FE:math.inf, # +INFINITY0x07FF:math.nan, # NaN (Not a Number)0x0800:math.nan, # NRes (Not at this Resolution)0x0801:math.nan, # Reserved for future0x0802:-math.inf # -INFINITY}mantissa = sfloat_value & 0x0FFFexponent = sfloat_value >> 12if (exponent >= 0x0008):exponent = -((0x000F + 1) - exponent)output = 0if (mantissa >= 0x07FE and mantissa <= 0x0802):output = reserved_float_values[mantissa]else:if (mantissa >= 0x0800):mantissa = -((0x0FFF + 1) - mantissa)magnitude = pow(10.0, exponent)output = (mantissa * magnitude)return outputlength = len(glucose_measurement_context)flags = glucose_measurement_context[length - 1]extflags_len = 1 if (flags & 0x80) else 0carb_len = 3 if (flags & 0x01) else 0if (flags & 0x01):	N/A

		oic.r.glucose.carb.carb = ieee11073_Sfloat_2_Float(glucose_measurement_context[length - carb_len - extflags_len - 3 : length - 1 - extflags_len - 3]) * 1000	
glucose_measurement_context[length - 1 - extflags_len - 3]	oic.r.glucose.carb	length = len(glucose_measurement_context)flags = glucose_measurement_context[length - 1]extflags_len = 1 if (flags & 0x80) else 0if (flags & 0x01): meal = { 1:"breakfast", 2:"lunch", 3:"dinner", 4:"snack", 5:"drink", 6:"supper", 7:"brunch" } oic.r.glucose.carb.meal = meal[glucose_measurement_context[length - 1 - extflags_len - 3]]	N/A
glucose_measurement_context[length - 2 - health_len - meal_len - carb_len - extflags_len - 3]	oic.r.glucose.exercise	length = len(glucose_measurement_context)flags = glucose_measurement_context[length - 1]extflags_len = 1 if (flags & 0x80) else 0carb_len = 3 if (flags & 0x01) else 0meal_len = 1 if (flags & 0x02) else 0health_len = 1 if (flags & 0x04) else 0if (flags & 0x08): oic.r.glucose.exercise.exercise = glucose_measurement_context[length - 2 - health_len - meal_len - carb_len - extflags_len - 3]	N/A
glucose_measurement_context[length - hba1c_len - medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3]	oic.r.glucose.hba1c	def ieee11073_Sfloat_2_Float(sfloat_value):# reserved value for Infinity or NaN (Not a Number)reserved_float_values = {0x07FE:math.inf, # +INFINITY0x07FF:math.nan, # NaN (Not a Number)0x0800:math.nan, # NRes (Not at this Resolution)0x0801:math.nan, # Reserved for future0x0802:-math.inf # -INFINITY}mantissa = sfloat_value & 0x0FFFexponent = sfloat_value >> 12if (exponent >= 0x0008):exponent = -((0x000F + 1) - exponent)output = 0if (mantissa >= 0x07FE and mantissa <= 0x0802):output = reserved_float_values[mantissa]else:if (mantissa >= 0x0800):mantissa = -((0x0FFF + 1) - mantissa)magnitude = pow(10.0, exponent)output = (mantissa * magnitude)return outputlength = len(glucose_measurement_context)flags = glucose_measurement_context[length - 1]extflags_len = 1 if (flags & 0x80) else 0carb_len = 3 if (flags & 0x01) else 0meal_len = 1 if (flags & 0x02) else 0health_len = 1 if (flags & 0x04) else 0exercise_len = 3 if (flags & 0x08) else 0medication_len = 3 if (flags & 0x10) else 0hba1c_len = 2 if (flags & 0x40) else 0if (flags & 0x40): oic.r.glucose.hba1c.hba1c = ieee11073_Sfloat_2_Float(glucose_measurement_context[length - hba1c_len - medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3])	N/A
glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3]	oic.r.glucose.health	length = len(glucose_measurement_context)flags = glucose_measurement_context[length - 1]extflags_len = 1 if (flags & 0x80) else 0carb_len = 3 if (flags & 0x01) else 0meal_len = 1 if (flags & 0x02) else 0health_len = 1 if (flags & 0x04) else 0if (flags & 0x04): health = { 1:"minor", 2:"major", 3:"menses", 4:"stress", 5:"none" } oic.r.glucose.health.health = health[glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3] & 0xf0] tester = { 1:"self", 2:"hcp", 3:"lab" } oic.r.glucose.tester.tester = tester[glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3] & 0x0f]	N/A

glucose_measurement_context[length - meal_len - carb_len - extflags_len - 3]	oic.r.glucose.meal	length = len(glucose_measurement_context)flags = glucose_measurement_context[length - 1]extflags_len = 1 if (flags & 0x80) else 0carb_len = 3 if (flags & 0x01) else 0meal_len = 1 if (flags & 0x02) else 0if (flags & 0x02): meal = { 1:"preprandial", 2:"postprandial", 3:"fasting", 4:"casual", 5:"bedtime" } oic.r.glucose.meal.meal = meal[glucose_measurement_context[length - meal_len - carb_len - extflags_len - 3]]	N/A
glucose_measurement_context[length - medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - 1 - exercise_len - health_len - meal_len - carb_len - extflags_len - 3]	oic.r.glucose.medication	def ieee11073_Sfloat_2_Float(sfloat_value):# reserved value for Infinity or NaN (Not a Number)reserved_float_values = {0x07FE:math.inf, # +INFINITY0x07FF:math.nan, # NaN (Not a Number)0x0800:math.nan, # NRes (Not at this Resolution)0x0801:math.nan, # Reserved for future0x0802:-math.inf # -INFINITY}mantissa = sfloat_value & 0x0FFFexponent = sfloat_value >> 12if (exponent >= 0x0008):exponent = -((0x000F + 1) - exponent)output = 0if (mantissa >= 0x07FE and mantissa <= 0x0802):output = reserved_float_values[mantissa]else:if (mantissa >= 0x0800):mantissa = -((0x0FFF + 1) - mantissa)magnitude = pow(10.0, exponent)output = (mantissa * magnitude)return outputlength = len(glucose_measurement_context)flags = glucose_measurement_context[length - 1]extflags_len = 1 if (flags & 0x80) else 0carb_len = 3 if (flags & 0x01) else 0meal_len = 1 if (flags & 0x02) else 0health_len = 1 if (flags & 0x04) else 0exercise_len = 3 if (flags & 0x08) else 0medication_len = 3 if (flags & 0x10) else 0hba1c_len = 2 if (flags & 0x40) else 0if (flags & 0x10): medication = ieee11073_Sfloat_2_Float(glucose_measurement_context[length - medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - 1 - exercise_len - health_len - meal_len - carb_len - extflags_len - 3]) oic.r.glucose.medication.medication = medication * 1000 oic.r.glucose.medication.units = "mL" if (flags & 0x20) else "mg"	N/A
glucose_measurement_context[length - 1 - exercise_len - health_len - meal_len - carb_len - extflags_len - 3]	oic.r.glucose.medication	length = len(glucose_measurement_context)flags = glucose_measurement_context[length - 1]extflags_len = 1 if (flags & 0x80) else 0carb_len = 3 if (flags & 0x01) else 0meal_len = 1 if (flags & 0x02) else 0health_len = 1 if (flags & 0x04) else 0exercise_len = 3 if (flags & 0x08) else 0if (flags & 0x10): regimen = { 1:"rapidacting", 2:"shortacting", 3:"intermediateacting", 4:"longacting", 5:"premix" } oic.r.glucose.medication.regimen = regimen[glucose_measurement_context[length - 1 - exercise_len - health_len - meal_len - carb_len - extflags_len - 3]]	N/A

Table 16 provides the details of the Properties that are part of "org.bluetooth.characteristic.glucose_measurement_context".

Table 16 – The Properties of "org.bluetooth.characteristic.glucose_measurement_context".

BLE Property name	Type	Required	Description
glucose_measurement_context[length - carb_len - extflags_len - 3 : length - 1 - extflags_len - 3]		no	Carbohydrate

glucose_measurement_context[length - 1 - extflags_len - 3]		no	Carbohydrate ID
glucose_measurement_context[length - 2 - health_len - meal_len - carb_len - extflags_len - 3]		no	Exercise Intensity
glucose_measurement_context[length - hba1c_len - medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3]		no	HbA1c
glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3]		no	Health, Tester
glucose_measurement_context[length - meal_len - carb_len - extflags_len - 3]		no	Meal
glucose_measurement_context[length - medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - 1 - exercise_len - health_len - meal_len - carb_len - extflags_len - 3]		no	Medication
glucose_measurement_context[length - 1 - exercise_len - health_len - meal_len - carb_len - extflags_len - 3]		no	Medication ID

9.3.3 Derived model definition

```
{
        "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.GLP.json#",
        "$schema": "http://json-schema.org/draft-04/schema#",
        "description" : "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
reserved.",
        "title": "Glucose",
        "definitions": {
        "byte": {
            "type": "integer",
            "minimum": 0,
            "maximum": 255
        },
        "byteArray": {
            "type": "array",
            "items": { "$ref": "#/definitions/byte" },
            "minItems": 1,
            "uniqueItems": false
        },
                "org.bluetooth.characteristic.glucose_measurement": {
                        "type": "object",
                        "properties": {
                 "glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len -
10]": {
                    "$ref": "#/definitions/byteArray",
                    "description": "Glucose Concentration",
                    "x-ocf-conversion": {
                        "x-ocf-alias": "oic.r.glucose",
                        "x-to-ocf": [
                            "def ieee11073_Sfloat_2_Float(sfloat_value):",
                                "# reserved value for Infinity or NaN (Not a Number)",
                                "reserved_float_values = {",
                                    "0x07FE:math.inf, # +INFINITY",
                                    "0x07FF:math.nan, # NaN (Not a Number)",
                                    "0x0800:math.nan, # NRes (Not at this Resolution)",
```



```
                                    "0x0801:math.nan, # Reserved for future",
                                    "0x0802:-math.inf # -INFINITY",
                                "}",
                                "mantissa = sfloat_value & 0x0FFF",
                                "exponent = sfloat_value >> 12",
                                "if (exponent >= 0x0008):",
                                    "exponent = -((0x000F + 1) - exponent)",
                                "output = 0",
                                "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
                                    "output = reserved_float_values[mantissa]",
                                "else:",
                                    "if (mantissa >= 0x0800):",
                                        "mantissa = -((0x0FFF + 1) - mantissa)",
                                    "magnitude = pow(10.0, exponent)",
                                    "output = (mantissa * magnitude)",
                                "return output",
                            "length = len(glucose_measurement)",
                            "flags = glucose_measurement[length - 1]",
                            "timeoffset_len = 2 if (flags & 0x01) else 0",
                            "if (flags & 0x02) == True:",
                            "   glucose = ieee11073_Sfloat_2_Float(glucose_measurement[length - 2 -
timeoffset_len - 10 : length - timeoffset_len - 10])",
                            "   oic.r.glucose.glucose = (glucose * 1000) if (flags & 0x04) else
(glucose * 0.1 * 1000 * 1000)",
                            "   oic.r.glucose.units = 'mmol/L' if (flags & 0x04) else 'mg/dL'",
                            "if (flags & 0x02) == False:",
                            "   oic.r.glucose.glucose = 0",
                            "   oic.r.glucose.units = 'mmol/L'"
                        ],
                        "x-from-ocf": [
                            "N/A"
                        ]
                    }
                },
                "glucose_measurement[length - 1 - 2 - timeoffset_len - 10]": {
                    "$ref": "#/definitions/byteArray",
                    "description": "Sample Location",
                    "x-ocf-conversion": {
                        "x-ocf-alias": "oic.r.glucose.samplelocation",
                        "x-to-ocf": [
                            "length = len(glucose_measurement)",
                            "flags = glucose_measurement[length - 1]",
                            "timeoffset_len = 2 if (flags & 0x01) else 0",
                            "if (flags & 0x02):",
                            "   samplelocation = { 1:'finger', 2:'ast', 3:'earlobe',
4:'ctrlsolution' }",
                            "   oic.r.glucose.samplelocation.samplelocation =
samplelocation[glucose_measurement[length - 1 - 2 - timeoffset_len - 10] & 0xf0]"
                        ],
                        "x-from-ocf": [
                            "N/A"
                        ]
                    }
                }
                       }
                },

                "org.bluetooth.characteristic.glucose_measurement_context": {
                        "type": "object",
            "properties": {
                "glucose_measurement_context[length - carb_len - extflags_len - 3 : length - 1 -
extflags_len - 3]": {
                    "$ref": "#/definitions/byteArray",
                    "description": "Carbohydrate",
                    "x-ocf-conversion": {
                        "x-ocf-alias": "oic.r.glucose.carb",
                        "x-to-ocf": [
                            "def ieee11073_Sfloat_2_Float(sfloat_value):",
                                "# reserved value for Infinity or NaN (Not a Number)",
                                "reserved_float_values = {",
                                    "0x07FE:math.inf, # +INFINITY",
```



```
                                    "0x07FF:math.nan, # NaN (Not a Number)",
                                    "0x0800:math.nan, # NRes (Not at this Resolution)",
                                    "0x0801:math.nan, # Reserved for future",
                                    "0x0802:-math.inf # -INFINITY",
                                "}",
                                "mantissa = sfloat_value & 0x0FFF",
                                "exponent = sfloat_value >> 12",
                                "if (exponent >= 0x0008):",
                                    "exponent = -((0x000F + 1) - exponent)",
                                "output = 0",
                                "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
                                    "output = reserved_float_values[mantissa]",
                                "else:",
                                    "if (mantissa >= 0x0800):",
                                        "mantissa = -((0x0FFF + 1) - mantissa)",
                                    "magnitude = pow(10.0, exponent)",
                                    "output = (mantissa * magnitude)",
                                "return output",
                            "length = len(glucose_measurement_context)",
                            "flags = glucose_measurement_context[length - 1]",
                            "extflags_len = 1 if (flags & 0x80) else 0",
                            "carb_len = 3 if (flags & 0x01) else 0",
                            "if (flags & 0x01): ",
                            "   oic.r.glucose.carb.carb =
ieee11073_Sfloat_2_Float(glucose_measurement_context[length - carb_len - extflags_len - 3 : length
- 1 - extflags_len - 3]) * 1000"
                        ],
                        "x-from-ocf": [
                            "N/A"
                        ]
                    }
                },
                "glucose_measurement_context[length - 1 - extflags_len - 3]": {
                    "$ref": "#/definitions/byteArray",
                    "description": "Carbohydrate ID",
                    "x-ocf-conversion": {
                        "x-ocf-alias": "oic.r.glucose.carb",
                        "x-to-ocf": [
                            "length = len(glucose_measurement_context)",
                            "flags = glucose_measurement_context[length - 1]",
                            "extflags_len = 1 if (flags & 0x80) else 0",
                            "if (flags & 0x01): ",
                            "   meal = { 1:'breakfast', 2:'lunch', 3:'dinner', 4:'snack',
5:'drink', 6:'supper', 7:'brunch' }",
                            "   oic.r.glucose.carb.meal = meal[glucose_measurement_context[length -
1 - extflags_len - 3]]"
                        ],
                        "x-from-ocf": [
                            "N/A"
                        ]
                    }
                },
                "glucose_measurement_context[length - 2 - health_len - meal_len - carb_len -
extflags_len - 3]": {
                    "$ref": "#/definitions/byteArray",
                    "description": "Exercise Intensity",
                    "x-ocf-conversion": {
                        "x-ocf-alias": "oic.r.glucose.exercise",
                        "x-to-ocf": [
                            "length = len(glucose_measurement_context)",
                            "flags = glucose_measurement_context[length - 1]",
                            "extflags_len = 1 if (flags & 0x80) else 0",
                            "carb_len = 3 if (flags & 0x01) else 0",
                            "meal_len = 1 if (flags & 0x02) else 0",
                            "health_len = 1 if (flags & 0x04) else 0",
                            "if (flags & 0x08): ",
                            "   oic.r.glucose.exercise.exercise =
glucose_measurement_context[length - 2 - health_len - meal_len - carb_len - extflags_len - 3]"
                        ],
                        "x-from-ocf": [
                            "N/A"
```



```
                        ]
                    }
                },
                "glucose_measurement_context[length - hba1c_len - medication_len - exercise_len -
health_len - meal_len - carb_len - extflags_len - 3 : length - medication_len - exercise_len -
health_len - meal_len - carb_len - extflags_len - 3]": {
                    "$ref": "#/definitions/byteArray",
                    "description": "HbA1c",
                    "x-ocf-conversion": {
                        "x-ocf-alias": "oic.r.glucose.hba1c",
                        "x-to-ocf": [
                            "def ieee11073_Sfloat_2_Float(sfloat_value):",
                                "# reserved value for Infinity or NaN (Not a Number)",
                                "reserved_float_values = {",
                                    "0x07FE:math.inf, # +INFINITY",
                                    "0x07FF:math.nan, # NaN (Not a Number)",
                                    "0x0800:math.nan, # NRes (Not at this Resolution)",
                                    "0x0801:math.nan, # Reserved for future",
                                    "0x0802:-math.inf # -INFINITY",
                                "}",
                                "mantissa = sfloat_value & 0x0FFF",
                                "exponent = sfloat_value >> 12",
                                "if (exponent >= 0x0008):",
                                    "exponent = -((0x000F + 1) - exponent)",
                                "output = 0",
                                "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
                                    "output = reserved_float_values[mantissa]",
                                "else:",
                                    "if (mantissa >= 0x0800):",
                                        "mantissa = -((0x0FFF + 1) - mantissa)",
                                    "magnitude = pow(10.0, exponent)",
                                    "output = (mantissa * magnitude)",
                                "return output",
                            "length = len(glucose_measurement_context)",
                            "flags = glucose_measurement_context[length - 1]",
                            "extflags_len = 1 if (flags & 0x80) else 0",
                            "carb_len = 3 if (flags & 0x01) else 0",
                            "meal_len = 1 if (flags & 0x02) else 0",
                            "health_len = 1 if (flags & 0x04) else 0",
                            "exercise_len = 3 if (flags & 0x08) else 0",
                            "medication_len = 3 if (flags & 0x10) else 0",
                            "hba1c_len = 2 if (flags & 0x40) else 0",
                            "if (flags & 0x40):",
                            "   oic.r.glucose.hba1c.hba1c =
ieee11073_Sfloat_2_Float(glucose_measurement_context[length - hba1c_len - medication_len -
exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - medication_len -
exercise_len - health_len - meal_len - carb_len - extflags_len - 3])"
                        ],
                        "x-from-ocf": [
                            "N/A"
                        ]
                    }
                },
                "glucose_measurement_context[length - health_len - meal_len - carb_len -
extflags_len - 3]": {
                    "$ref": "#/definitions/byteArray",
                    "description": "Health, Tester",
                    "x-ocf-conversion": {
                        "x-ocf-alias": "oic.r.glucose.health",
                        "x-to-ocf": [
                            "length = len(glucose_measurement_context)",
                            "flags = glucose_measurement_context[length - 1]",
                            "extflags_len = 1 if (flags & 0x80) else 0",
                            "carb_len = 3 if (flags & 0x01) else 0",
                            "meal_len = 1 if (flags & 0x02) else 0",
                            "health_len = 1 if (flags & 0x04) else 0",
                            "if (flags & 0x04): ",
                            "   health = { 1:'minor', 2:'major', 3:'menses', 4:'stress',
5:'none' }",
                            "   oic.r.glucose.health.health =
health[glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3] &
```



```
0xf0]",
                            "   tester = { 1:'self', 2:'hcp', 3:'lab' }",
                            "   oic.r.glucose.tester.tester =
tester[glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3] &
0x0f]"
                        ],
                        "x-from-ocf": [
                            "N/A"
                        ]
                    }
                },
                "glucose_measurement_context[length - meal_len - carb_len - extflags_len - 3]": {
                    "$ref": "#/definitions/byteArray",
                    "description": "Meal",
                    "x-ocf-conversion": {
                        "x-ocf-alias": "oic.r.glucose.meal",
                        "x-to-ocf": [
                            "length = len(glucose_measurement_context)",
                            "flags = glucose_measurement_context[length - 1]",
                            "extflags_len = 1 if (flags & 0x80) else 0",
                            "carb_len = 3 if (flags & 0x01) else 0",
                            "meal_len = 1 if (flags & 0x02) else 0",
                            "if (flags & 0x02): ",
                            "   meal = { 1:'preprandial', 2:'postprandial', 3:'fasting',
4:'casual', 5:'bedtime' }",
                            "   oic.r.glucose.meal.meal = meal[glucose_measurement_context[length -
meal_len - carb_len - extflags_len - 3]]"
                        ],
                        "x-from-ocf": [
                            "N/A"
                        ]
                    }
                },
                "glucose_measurement_context[length - medication_len - exercise_len - health_len -
meal_len - carb_len - extflags_len - 3 : length - 1 - exercise_len - health_len - meal_len -
carb_len - extflags_len - 3]": {
                    "$ref": "#/definitions/byteArray",
                    "description": "Medication",
                    "x-ocf-conversion": {
                        "x-ocf-alias": "oic.r.glucose.medication",
                        "x-to-ocf": [
                            "def ieee11073_Sfloat_2_Float(sfloat_value):",
                                "# reserved value for Infinity or NaN (Not a Number)",
                                "reserved_float_values = {",
                                    "0x07FE:math.inf, # +INFINITY",
                                    "0x07FF:math.nan, # NaN (Not a Number)",
                                    "0x0800:math.nan, # NRes (Not at this Resolution)",
                                    "0x0801:math.nan, # Reserved for future",
                                    "0x0802:-math.inf # -INFINITY",
                                "}",
                                "mantissa = sfloat_value & 0x0FFF",
                                "exponent = sfloat_value >> 12",
                                "if (exponent >= 0x0008):",
                                    "exponent = -((0x000F + 1) - exponent)",
                                "output = 0",
                                "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
                                    "output = reserved_float_values[mantissa]",
                                "else:",
                                    "if (mantissa >= 0x0800):",
                                        "mantissa = -((0x0FFF + 1) - mantissa)",
                                    "magnitude = pow(10.0, exponent)",
                                    "output = (mantissa * magnitude)",
                                "return output",
                            "length = len(glucose_measurement_context)",
                            "flags = glucose_measurement_context[length - 1]",
                            "extflags_len = 1 if (flags & 0x80) else 0",
                            "carb_len = 3 if (flags & 0x01) else 0",
                            "meal_len = 1 if (flags & 0x02) else 0",
                            "health_len = 1 if (flags & 0x04) else 0",
                            "exercise_len = 3 if (flags & 0x08) else 0",
                            "medication_len = 3 if (flags & 0x10) else 0",
```



```
                            "hba1c_len = 2 if (flags & 0x40) else 0",
                            "if (flags & 0x10): ",
                            "   medication =
ieee11073_Sfloat_2_Float(glucose_measurement_context[length - medication_len - exercise_len -
health_len - meal_len - carb_len - extflags_len - 3 : length - 1 - exercise_len - health_len -
meal_len - carb_len - extflags_len - 3])",
                            "   oic.r.glucose.medication.medication = medication * 1000",
                            "   oic.r.glucose.medication.units = 'mL' if (flags & 0x20) else 'mg'"
                        ],
                        "x-from-ocf": [
                            "N/A"
                        ]
                    }
                },
                "glucose_measurement_context[length - 1 - exercise_len - health_len - meal_len -
carb_len - extflags_len - 3]": {
                    "$ref": "#/definitions/byteArray",
                    "description": "Medication ID",
                    "x-ocf-conversion": {
                        "x-ocf-alias": "oic.r.glucose.medication",
                        "x-to-ocf": [
                            "length = len(glucose_measurement_context)",
                            "flags = glucose_measurement_context[length - 1]",
                            "extflags_len = 1 if (flags & 0x80) else 0",
                            "carb_len = 3 if (flags & 0x01) else 0",
                            "meal_len = 1 if (flags & 0x02) else 0",
                            "health_len = 1 if (flags & 0x04) else 0",
                            "exercise_len = 3 if (flags & 0x08) else 0",
                            "if (flags & 0x10): ",
                            "   regimen = { 1:'rapidacting', 2:'shortacting',
3:'intermediateacting', 4:'longacting', 5:'premix' }",
                            "   oic.r.glucose.medication.regimen =
regimen[ glucose_measurement_context[length - 1 - exercise_len - health_len - meal_len - carb_len -
extflags_len - 3] ]"
                        ],
                        "x-from-ocf": [
                            "N/A"
                        ]
                    }
                }
            }
        },

        "type": "object",

        "allOf": [
                { "$ref": "#/definitions/byte" },
                { "$ref": "#/definitions/byteArray" },
        { "$ref":
"#/definitions/org.bluetooth.characteristic.org.bluetooth.characteristic.glucose_measurement" },
        { "$ref":
"#/definitions/org.bluetooth.characteristic.org.bluetooth.characteristic.glucose_measurement_contex
t" }
        ],

        "required": [
                "glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len -
10]"
        ]
}
```

9.4 Health Thermometer Mapping

9.4.1 Derived model

The derived model: "org.bluetooth.characteristic.temperature_measurement".

9.4.2 Property definition

Table 17 provides the detailed per Property mapping for "org.bluetooth.characteristic.temperature_measurement".

Table 17 – The Property mapping for "org.bluetooth.characteristic.temperature_measurement".

BLE Property name	OCF Resource	To OCF	From OCF
temperature_measurement[length - 5 : length - 1]	oic.r.temperature	# convert IEEE11073 FLOAT to floatdef ieee11073_Float_2_Float(float_value):# reserved value for Infinity or NaN (Not a Number)reserved_float_values = {0x007FFFFE:math.inf, # +INFINITY0x007FFFFF:math.nan, # NaN (Not a Number)0x00800000:math.nan, # NRes (Not at this Resolution)0x00800001:math.nan, # Reserved for future0x00800002:-math.inf # -INFINITY}mantissa = float_value & 0x00FFFFFFexponent = float_value >> 24if (exponent >= 0x00000080):exponent = -((0x000000FF + 1) - exponent)output = 0if (mantissa >= 0x007FFFFE and mantissa <= 0x00800002):output = reserved_float_values[mantissa]else:if (mantissa >= 0x00800000):mantissa = -((0x00FFFFFF + 1) - mantissa)magnitude = pow(10.0, exponent)output = (mantissa * magnitude)return outputlength = len(temperature_measurement)flags = temperature_measurement[length - 1]oic.r.temperature.temperature = ieee11073_Float_2_Float(temperature_measurement[length - 5 : length - 1])oic.r.temperature.units = 'F' if (flags & 0x01) else 'C'	N/A
temperature_measurement[length - temperaturetype_len - timestamp_len - 5]	oic.r.body.location.temperature	length = len(temperature_measurement)flags = temperature_measurement[length - 1]timestamp_len = 7 if (flags & 0x02) 0temperaturetype_len = 1 if (flags & 0x04) 0if (flags & 0x04): bloc = { 1:'xxx', 2:'body', 3:'ear', 4:'finger', 5:'gastro', 6:'mouth', 7:'rectum', 8:'toe', 9:'tympanum' } oic.r.body.location.temperature.bloc = bloc[temperature_measurement[length - temperaturetype_len - timestamp_len - 5]]	N/A

Table 18 provides the details of the Properties that are part of "org.bluetooth.characteristic.temperature_measurement".

Table 18 – The Properties of "org.bluetooth.characteristic.temperature_measurement".

BLE Property name	Type	Required	Description
temperature_measurement[length - 5 : length - 1]		yes	Temperature
temperature_measurement[length - temperaturetype_len - timestamp_len - 5]		no	Temperature Type

9.4.3 Derived model definition

```
{
        "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.HTP.json#",
        "$schema": "http://json-schema.org/draft-04/schema#",
        "description" : "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
reserved.",
        "title": "Health Thermometer",
        "definitions": {
                "byte": {
                        "type": "integer",
            "minimum": 0,
            "maximum": 255
                },
                "byteArray": {
                        "type": "array",
            "items": { "$ref": "#/definitions/byte" },
            "minItems": 1,
            "uniqueItems": false
                },
                "org.bluetooth.characteristic.temperature_measurement" : {
                        "type": "object",
                        "properties" : {
                                "temperature_measurement[ length - 5 : length - 1 ]" : {
                                        "$ref": "#/definitions/byteArray",
                                        "description": "Temperature",
                                        "x-ocf-conversion": {
                                                "x-ocf-alias": "oic.r.temperature",
                                                "x-to-ocf": [
                                                        "# convert IEEE11073 FLOAT to float",
                                                        "def ieee11073_Float_2_Float(float_value):",
                                                            "# reserved value for Infinity or NaN
(Not a Number)",
                                                            "reserved_float_values = {",
                                                                "0x007FFFFE:math.inf, # +INFINITY",
                                                                "0x007FFFFF:math.nan, # NaN (Not a
Number)",
                                                                "0x00800000:math.nan, # NRes (Not at
this Resolution)",
                                                                "0x00800001:math.nan, # Reserved for
future",
                                                                "0x00800002:-math.inf # -INFINITY",
                                                            "}",
                                                            "mantissa = float_value & 0x00FFFFFF",
                                                            "exponent = float_value >> 24",
                                                            "if (exponent >= 0x00000080):",
                                                                "exponent = -((0x000000FF + 1) -
exponent)",
                                                            "output = 0",
                                                            "if (mantissa >= 0x007FFFFE and mantissa
<= 0x00800002):",
                                                                "output =
reserved_float_values[mantissa]",
                                                            "else:",
                                                                "if (mantissa >= 0x00800000):",
                                                                    "mantissa = -((0x00FFFFFF + 1) -
mantissa)",
                                                                "magnitude = pow(10.0, exponent)",
                                                                "output = (mantissa * magnitude)",
                                                            "return output",
                                                        "length = len(temperature_measurement)",
                                                        "flags = temperature_measurement[length -
1]",
                                                        "oic.r.temperature.temperature =
ieee11073_Float_2_Float(temperature_measurement[ length - 5 : length - 1 ])",
                                                        "oic.r.temperature.units = 'F' if (flags &
0x01) else 'C'"
                                                ],
                                                "x-from-ocf": [
                                                        "N/A"
```

```
                                                ]
                                        }
                                },
                                "temperature_measurement[ length - temperaturetype_len -
timestamp_len - 5 ]" : {
                                        "$ref": "#/definitions/byteArray",
                                        "description": "Temperature Type",
                                        "x-ocf-conversion": {
                                                "x-ocf-alias": "oic.r.body.location.temperature",
                                                "x-to-ocf": [
                                                        "length = len(temperature_measurement)",
                                                        "flags = temperature_measurement[length -
1]",
                                                        "timestamp_len = 7 if (flags & 0x02) 0",
                                                        "temperaturetype_len = 1 if (flags & 0x04)
0",
                                                        "if (flags & 0x04):",
                                                        "   bloc = { 1:'xxx', 2:'body', 3:'ear',
4:'finger', 5:'gastro', 6:'mouth', 7:'rectum', 8:'toe', 9:'tympanum' }",
                                                        "   oic.r.body.location.temperature.bloc =
bloc[temperature_measurement[ length - temperaturetype_len - timestamp_len - 5 ] ]"
                                                ],
                                                "x-from-ocf": [
                                                        "N/A"
                                                ]
                                        }
                                }
                        }
                },

                "type": "object",

                "allOf": [
                        { "$ref": "#/definitions/byte" },
                        { "$ref": "#/definitions/byteArray" },
                { "$ref": "#/definitions/org.bluetooth.characteristic.temperature_measurement" }
                ],

                "required": [
                        "temperature_measurement[ length - 5 : length - 1 ]"
                ]
}
```

9.5 Weight Scale Mapping

9.5.1 Derived model

The derived model: "org.bluetooth.characteristic.weight_measurement".

The derived model: "org.bluetooth.characteristic.body_composition_measurement".

9.5.2 Property definition

Table 19 provides the detailed per Property mapping for "org.bluetooth.characteristic.weight_measurement".

Table 19 – The Property mapping for "org.bluetooth.characteristic.weight_measurement".

BLE Property name	OCF Resource	To OCF	From OCF
weight_measurement[length - 3 : length - 1]	oic.r.weight	length = len(weight_measurement)flags = weight_measurement[length - 1]timeoffset_len = 7 if (flags & 0x02) else 0oic.r.weight.weight = int.from_bytes(weight_measurement[length - 3 : length - 1], 'big')oic.r.weight.units = 'lb' if (flags & 0x01) else 'kg'	N/A

weight_measurement[length - 2 - userid_len - timeoffset_len - 3 : length - userid_len - timeoffset_len - 3]	oic.r.bmi	length = len(weight_measurement)flags = weight_measurement[length - 1]timeoffset_len = 7 if (flags & 0x02) else 0userid_len = 1 if (flags & 0x04) else 0if (flags & 0x08): oic.r.bmi.bmi = int.from_bytes(weight_measurement[length - 2 - userid_len - timeoffset_len - 3 : length - userid_len - timeoffset_len - 3], 'big')	N/A
weight_measurement[length - height_len - userid_len - timeoffset_len - 3 : length - 2 - userid_len - timeoffset_len - 3]	oic.r.height	length = len(weight_measurement)flags = weight_measurement[length - 1]timeoffset_len = 7 if (flags & 0x02) else 0userid_len = 1 if (flags & 0x04) else 0height_len = 4 if (flags & 0x08) else 0if (flags & 0x08): oic.r.height.height = int.from_bytes(weight_measurement[length - height_len - userid_len - timeoffset_len - 3 : length - 2 - userid_len - timeoffset_len - 3], 'big') oic.r.height.units = 'in' if (flags & 0x01) else 'm'	N/A

Table 20 provides the details of the Properties that are part of "org.bluetooth.characteristic.weight_measurement".

Table 20 – The Properties of "org.bluetooth.characteristic.weight_measurement".

BLE Property name	Type	Required	Description
weight_measurement[length - 3 : length - 1]		yes	Weight
weight_measurement[length - 2 - userid_len - timeoffset_len - 3 : length - userid_len - timeoffset_len - 3]		no	BMI
weight_measurement[length - height_len - userid_len - timeoffset_len - 3 : length - 2 - userid_len - timeoffset_len - 3]		no	Height

Table 21 provides the detailed per Property mapping for "org.bluetooth.characteristic.body_composition_measurement".

Table 21 – The Property mapping for "org.bluetooth.characteristic.body_composition_measurement".

BLE Property name	OCF Resource	To OCF	From OCF
body_composition_measurement[length - 4 : length - 2]	oic.r.body.fat	length = len(body_composition_measurement)oic.r.body.fat.bodyfat = int.from_bytes(body_composition_measurement[length - 4 : length - 2], 'big')oic.r.body.fat.units = '%'	N/A
body_composition_measurement[length - bwm_len - slm_len - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4]	oic.r.body.water	length = len(body_composition_measurement)flags_upperbyte = body_composition_measurement[length - 2]flags_lowerbyte = body_composition_measurement[length - 1]timestamp_len = 7 if (flags_lowerbyte & 0x02) else 0userid_len = 1 if (flags_lowerbyte & 0x04) else 0basal_len = 2 if (flags_lowerbyte & 0x08) else 0muscle_len = 2 if (flags_lowerbyte & 0x10) else 0mm_len = 2 if (flags_lowerbyte & 0x20) else	N/A

		0ffm_len = 2 if (flags_lowerbyte & 0x40) else 0slm_len = 2 if (flags_lowerbyte & 0x80) else 0bwm_len = 2 if (flags_upperbyte & 0x01) else 0if (flags_lowerbyte & 0x01): oic.r.body.water.bwater = int.from_bytes(body_composition_measurement[length - bwm_len - slm_len - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4], 'big') oic.r.body.water.units = 'lb' if (flags_lowerbyte & 0x01) 'kg'	
body_composition_measurement[length - slm_len - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4]	oic.r.body.slm	length = len(body_composition_measurement)flags_upperbyte = body_composition_measurement[length - 2]flags_lowerbyte = body_composition_measurement[length - 1]timestamp_len = 7 if (flags_lowerbyte & 0x02) else 0userid_len = 1 if (flags_lowerbyte & 0x04) else 0basal_len = 2 if (flags_lowerbyte & 0x08) else 0muscle_len = 2 if (flags_lowerbyte & 0x10) else 0mm_len = 2 if (flags_lowerbyte & 0x20) else 0ffm_len = 2 if (flags_lowerbyte & 0x40) else 0slm_len = 2 if (flags_lowerbyte & 0x80) else 0if (flags_lowerbyte & 0x01): oic.r.body.slm.bwater = int.from_bytes(body_composition_measurement[length - slm_len - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4], 'big') oic.r.body.slm.units = 'lb' if (flags_lowerbyte & 0x01) 'kg'	N/A
body_composition_measurement[length - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4]	oic.r.body.ffm	length = len(body_composition_measurement)flags_upperbyte = body_composition_measurement[length - 2]flags_lowerbyte = body_composition_measurement[length - 1]timestamp_len = 7 if (flags_lowerbyte & 0x02) else 0userid_len = 1 if (flags_lowerbyte & 0x04) else 0basal_len = 2 if (flags_lowerbyte & 0x08) else 0muscle_len = 2 if (flags_lowerbyte & 0x10) else 0mm_len = 2 if (flags_lowerbyte & 0x20) else 0ffm_len = 2 if (flags_lowerbyte & 0x40) else 0if (flags_lowerbyte & 0x01): oic.r.body.ffm.bwater = int.from_bytes(body_composition_measurement[length - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4], 'big') oic.r.body.ffm.units = 'lb' if (flags_lowerbyte & 0x01) 'kg'	N/A

Table 22 provides the details of the Properties that are part of "org.bluetooth.characteristic.body_composition_measurement".

Table 22 – The Properties of "org.bluetooth.characteristic.body_composition_measurement".

BLE Property name	Type	Required	Description
body_composition_measurement[length - 4 : length - 2]		no	Body Fat Percentage
body_composition_measurement[length - bwm_len - slm_len - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len - ffm_len - mm_len - muscle_len -		no	Body Water Mass

basal_len - userid_len - timestamp_len - 4]			
body_composition_measurement[length - slm_len - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4]		no	Soft Lean Mass
body_composition_measurement[length - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4]		no	Fat Free Mass

9.5.3 Derived model definition

```
{
        "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.WSS.json#",
        "$schema": "http://json-schema.org/draft-04/schema#",
        "description" : "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
reserved.",
        "title": "Weight Scale",
        "definitions": {
                "byte": {
                        "type": "integer",
            "minimum": 0,
            "maximum": 255
                },
                "byteArray": {
                        "type": "array",
            "items": { "$ref": "#/definitions/byte" },
            "minItems": 1,
            "uniqueItems": false
                },
                "org.bluetooth.characteristic.weight_measurement" : {
                        "type": "object",
                        "properties": {
                                "weight_measurement[length - 3 : length - 1]" : {
                                        "$ref": "#/definitions/byteArray",
                                        "description": "Weight",
                                        "x-ocf-conversion": {
                                                "x-ocf-alias": "oic.r.weight",
                                                "x-to-ocf": [
                                                        "length = len(weight_measurement)",
                                                        "flags = weight_measurement[length - 1]",
                                                        "timeoffset_len = 7 if (flags & 0x02) else
0",
                                                        "oic.r.weight.weight =
int.from_bytes(weight_measurement[length - 3 : length - 1], 'big')",
                                                        "oic.r.weight.units = 'lb' if (flags & 0x01)
else 'kg'"
                                                ],
                                                "x-from-ocf": [
                                                        "N/A"
                                                ]
                                        }
                                },
                                "weight_measurement[length - 2 - userid_len - timeoffset_len - 3 :
length - userid_len - timeoffset_len - 3]" : {
                                        "$ref": "#/definitions/byteArray",
                                        "description": "BMI",
                                        "x-ocf-conversion": {
                                                "x-ocf-alias": "oic.r.bmi",
                                                "x-to-ocf": [
                                                        "length = len(weight_measurement)",
                                                        "flags = weight_measurement[length - 1]",
                                                        "timeoffset_len = 7 if (flags & 0x02) else
0",
```

```
                                                        "userid_len = 1 if (flags & 0x04) else 0",
                                                        "if (flags & 0x08):",
                                                        "   oic.r.bmi.bmi =
int.from_bytes(weight_measurement[length - 2 - userid_len - timeoffset_len - 3 : length -
userid_len - timeoffset_len - 3], 'big')"
                                                ],
                                                "x-from-ocf": [
                                                        "N/A"
                                                ]
                                        }
                                },
                                "weight_measurement[length - height_len - userid_len -
timeoffset_len - 3 : length - 2 - userid_len - timeoffset_len - 3]" : {
                                        "$ref": "#/definitions/byteArray",
                                        "description": "Height",
                                        "x-ocf-conversion": {
                                                "x-ocf-alias": "oic.r.height",
                                                "x-to-ocf": [
                                                        "length = len(weight_measurement)",
                                                        "flags = weight_measurement[length - 1]",
                                                        "timeoffset_len = 7 if (flags & 0x02) else
0",
                                                        "userid_len = 1 if (flags & 0x04) else 0",
                                                        "height_len = 4 if (flags & 0x08) else 0",
                                                        "if (flags & 0x08):",
                                                        "   oic.r.height.height =
int.from_bytes(weight_measurement[length - height_len - userid_len - timeoffset_len - 3 : length -
2 - userid_len - timeoffset_len - 3], 'big')",
                                                        "   oic.r.height.units = 'in' if (flags &
0x01) else 'm'"
                                                ],
                                                "x-from-ocf": [
                                                        "N/A"
                                                ]
                                        }
                                }
                        }
                },

                "org.bluetooth.characteristic.body_composition_measurement" : {
                        "type": "object",
                        "properties": {
                                "body_composition_measurement[ length - 4 : length - 2]" : {
                                        "$ref": "#/definitions/byteArray",
                                        "description": "Body Fat Percentage",
                                        "x-ocf-conversion": {
                                                "x-ocf-alias": "oic.r.body.fat",
                                                "x-to-ocf": [
                                                        "length = len(body_composition_measurement)",
                                                        "oic.r.body.fat.bodyfat =
int.from_bytes(body_composition_measurement[ length - 4 : length - 2], 'big')",
                                                        "oic.r.body.fat.units = '%'"
                                                ],
                                                "x-from-ocf": [
                                                        "N/A"
                                                ]
                                        }
                                },
                                "body_composition_measurement[ length - bwm_len - slm_len - ffm_len
- mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len - ffm_len -
mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 ]" : {
                                        "$ref": "#/definitions/byteArray",
                                        "description": "Body Water Mass",
                                        "x-ocf-conversion": {
                                                "x-ocf-alias": "oic.r.body.water",
                                                "x-to-ocf": [
                                                        "length = len(body_composition_measurement)",
                                                        "flags_upperbyte =
body_composition_measurement[length - 2]",
                                                        "flags_lowerbyte =
body_composition_measurement[length - 1]",
```



```
                                                        "timestamp_len = 7 if (flags_lowerbyte &
    0x02) else 0",
                                                        "userid_len = 1 if (flags_lowerbyte & 0x04)
    else 0",
                                                        "basal_len = 2 if (flags_lowerbyte & 0x08)
    else 0",
                                                        "muscle_len = 2 if (flags_lowerbyte & 0x10)
    else 0",
                                                        "mm_len = 2 if (flags_lowerbyte & 0x20) else
    0",
                                                        "ffm_len = 2 if (flags_lowerbyte & 0x40) else
    0",
                                                        "slm_len = 2 if (flags_lowerbyte & 0x80) else
    0",
                                                        "bwm_len = 2 if (flags_upperbyte & 0x01) else
    0",
                                                        "if (flags_lowerbyte & 0x01): ",
                                                        "   oic.r.body.water.bwater =
    int.from_bytes(body_composition_measurement[ length - bwm_len - slm_len - ffm_len - mm_len -
    muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len - ffm_len - mm_len -
    muscle_len - basal_len - userid_len - timestamp_len - 4 ], 'big')",
                                                        "   oic.r.body.water.units = 'lb' if
    (flags_lowerbyte & 0x01) 'kg'"
                                                ],
                                                "x-from-ocf": [
                                                        "N/A"
                                                ]
                                        }
                                },
                                "body_composition_measurement[ length - slm_len - ffm_len - mm_len -
    muscle_len - basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len - muscle_len -
    basal_len - userid_len - timestamp_len - 4 ]" : {
                                        "$ref": "#/definitions/byteArray",
                                        "description": "Soft Lean Mass",
                                        "x-ocf-conversion": {
                                                "x-ocf-alias": "oic.r.body.slm",
                                                "x-to-ocf": [
                                                        "length = len(body_composition_measurement)",
                                                        "flags_upperbyte =
    body_composition_measurement[length - 2]",
                                                        "flags_lowerbyte =
    body_composition_measurement[length - 1]",
                                                        "timestamp_len = 7 if (flags_lowerbyte &
    0x02) else 0",
                                                        "userid_len = 1 if (flags_lowerbyte & 0x04)
    else 0",
                                                        "basal_len = 2 if (flags_lowerbyte & 0x08)
    else 0",
                                                        "muscle_len = 2 if (flags_lowerbyte & 0x10)
    else 0",
                                                        "mm_len = 2 if (flags_lowerbyte & 0x20) else
    0",
                                                        "ffm_len = 2 if (flags_lowerbyte & 0x40) else
    0",
                                                        "slm_len = 2 if (flags_lowerbyte & 0x80) else
    0",
                                                        "if (flags_lowerbyte & 0x01): ",
                                                        "   oic.r.body.slm.bwater =
    int.from_bytes(body_composition_measurement[ length - slm_len - ffm_len - mm_len - muscle_len -
    basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len - muscle_len - basal_len -
    userid_len - timestamp_len - 4 ], 'big')",
                                                        "   oic.r.body.slm.units = 'lb' if
    (flags_lowerbyte & 0x01) 'kg'"
                                                ],
                                                "x-from-ocf": [
                                                        "N/A"
                                                ]
                                        }
                                },
                                "body_composition_measurement[ length - ffm_len - mm_len -
    muscle_len - basal_len - userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len
```



```
- userid_len - timestamp_len - 4 ]" : {
                                        "$ref": "#/definitions/byteArray",
                                        "description": "Fat Free Mass",
                                        "x-ocf-conversion": {
                                                "x-ocf-alias": "oic.r.body.ffm",
                                                "x-to-ocf": [
                                                        "length = len(body_composition_measurement)",
                                                        "flags_upperbyte =
body_composition_measurement[length - 2]",
                                                        "flags_lowerbyte =
body_composition_measurement[length - 1]",
                                                        "timestamp_len = 7 if (flags_lowerbyte &
0x02) else 0",
                                                        "userid_len = 1 if (flags_lowerbyte & 0x04)
else 0",
                                                        "basal_len = 2 if (flags_lowerbyte & 0x08)
else 0",
                                                        "muscle_len = 2 if (flags_lowerbyte & 0x10)
else 0",
                                                        "mm_len = 2 if (flags_lowerbyte & 0x20) else
0",
                                                        "ffm_len = 2 if (flags_lowerbyte & 0x40) else
0",
                                                        "if (flags_lowerbyte & 0x01): ",
                                                        "   oic.r.body.ffm.bwater =
int.from_bytes(body_composition_measurement[ length - ffm_len - mm_len - muscle_len - basal_len -
userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len - userid_len -
timestamp_len - 4 ], 'big')",
                                                        "   oic.r.body.ffm.units = 'lb' if
(flags_lowerbyte & 0x01) 'kg'"
                                                ],
                                                "x-from-ocf": [
                                                        "N/A"
                                                ]
                                        }
                                }
                        }
                }
        },

        "type": "object",

        "allOf": [
                { "$ref": "#/definitions/byte" },
                { "$ref": "#/definitions/byteArray" },
        { "$ref": "#/definitions/org.bluetooth.characteristic.weight_measurement" },
        { "$ref": "#/definitions/org.bluetooth.characteristic.body_composition_measurement" }
        ],

        "required": [
                "weight_measurement[length - 3 : length - 1]"
        ]
}

```


Annex A
(Informative)
BLE GATT based Data Model

A.1 BLE GATT based data model & GATT features

A.1.1 Introduction

The Generic Attribute Profile (GATT) defines a service framework using the Attribute Protocol. This framework defines procedures and formats of services and their characteristics. The procedures defined include discovering, reading, writing, notifying and indicating characteristics, as well as configuring the broadcast of characteristics.

A.1.2 Profile dependency

Figure A-1 depicts the structure and the dependencies of the profiles. A profile is dependent upon another profile if it re-uses parts of that profile by implicitly or explicitly referencing it.

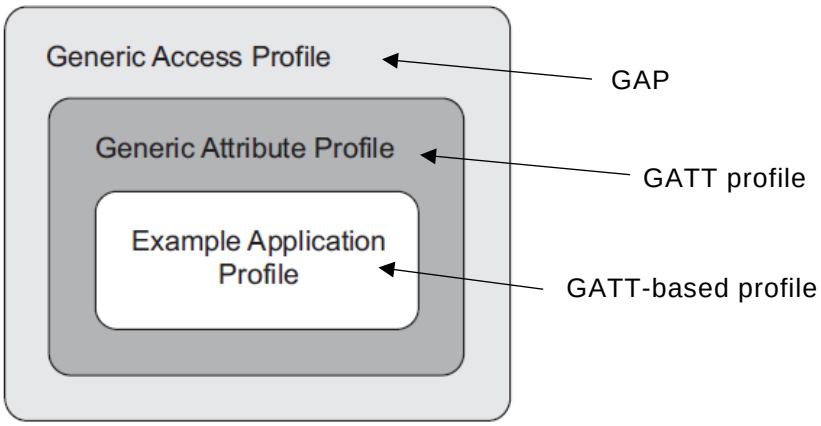


Figure A-1 – profile dependencies

A.1.3 Configurations and roles

There are two roles defined in GATT profile:

- Client: This is the device that initiates commands and requests towards the server and can receive responses, indications and notifications sent by the server.
- Server: This is the device that accepts incoming commands and requests from the client and sends responses, indications and notifications to a client.

A device can act in both roles at the same time.

A.1.4 GATT profile hierarchy

A.1.4.1 Introduction

The GATT Profile specifies the structure in which profile data is exchanged. This structure defines basic elements such as services and characteristics, used in a profile. All of the elements are contained by Attributes. Attributes used in the ATT are containers that carry this profile data.

The top level of the hierarchy is a profile. A profile is composed of one or more services necessary to fulfil a use case. A service is composed of characteristics or references to other services. Each characteristic contains a value and may contain optional information about the value. The service

and characteristic and the components of the characteristic (i.e. value and descriptors) contain the profile data and are all stored in Attributes on the server.

Under GATT profile, entity that provides Service-Characteristics data model plays “server” role and entity that gets data from GATT server plays “client” role.

There are other application profiles based on GATT profile. They are called “GATT-based profiles”.

Figure A-2 Illustrates the GATT profile hierarchy.

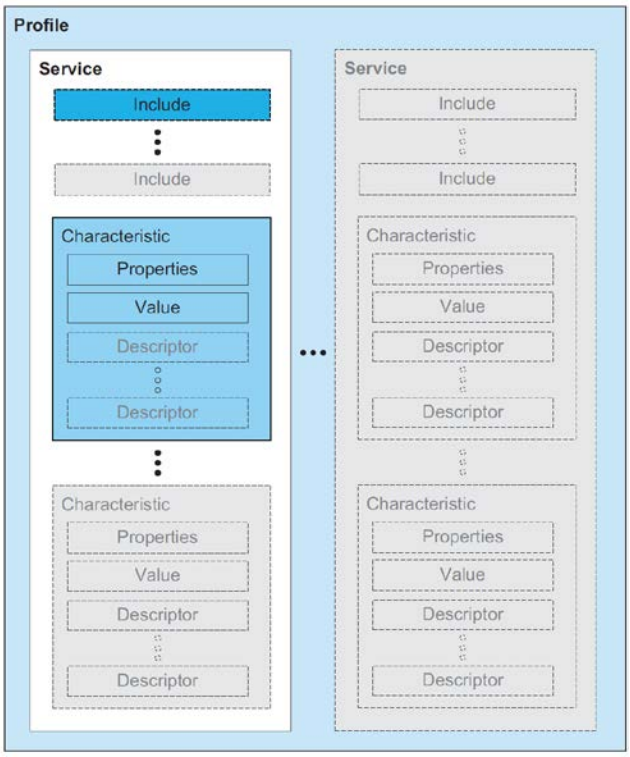


Figure A-2 – GATT profile hierarchy

A.1.4.2 Characteristic

A characteristic is a value used in a service along with properties and configuration information about how the value is accessed and information about how the value is displayed or represented. In GATT, a characteristic is defined by its characteristic definition. A characteristic definition contains a characteristic declaration, characteristic properties, and a value and may contain descriptors that describe the value or permit configuration of the server with respect to the characteristic.

A.1.4.3 GATT features

GATT profile also supports GATT features. GATT feature defines how GATT-based data exchanges take place. Each feature is mapped to one or more sub-procedures. These sub-procedures describe how the ATT is used to accomplish the corresponding feature, please see Table A-1.

Table A-1 – GATT Features and ATT protocol

	Feature	Sub-procedure	ATT protocol
1	Server Configuration	Exchange MTU	Exchange MTU Request

			Exchange MTU Response Error Response
2	Primary Service Discovery	Discover All Primary Services	Read By Group Type Request Read By Group Type Response Error Response
		Discover Primary Services by service UUID	Find By Type Value Request Find By Type Value Response Error Response
3	Relationship Discovery	Find Included Services	Read By Type Request Read By Type Response Error Response
4	Characteristic Discovery	Discover All Characteristic of a Service	Read By Type Request Read By Type Response Error Response
		Discover Characteristic by UUID	Read By Type Request Read By Type Response Error Response
5	Characteristic Descriptor Discovery	Discover All Characteristic Descriptors	Find Information Request Find Information Response Error Response
6	Characteristic Value Read	Read Characteristic Value	Read Request Read Response Error Response
		Read Using Characteristic UUID	Read By Type Request Read By Type Response Error Response
		Read Long Characteristic Values	Read Blob Request Read Blob Response Error Response
		Read Multiple Characteristic Values	Read Multiple Request Read Multiple Response Error Response
7	Characteristic Value Write	Write Without Response	Write Command
		Signed Write Without Response	Write Command
		Write Characteristic Value	Write Request Write Response Error Response
		Write Long Characteristic Values	Prepare Write Request Prepare Write Response Execute Write Request Execute Write Response Error Response
		Characteristic Value Reliable Writes	Prepare Write Request Prepare Write Response Execute Write Request Execute Write Response Error Response

8	Characteristic Value Notification	Notifications	Handle Value Notification
9	Characteristic Value Indication	Indications	Handle Value Indication Handle Value Confirmation
10	Characteristic Descriptor Value Read	Read Characteristic Descriptors	Read Request Read Response Error Response
		Read Long Characteristic Descriptors	Read Blob Request Read Blob Response Error Response
11	Characteristic Descriptor Value Write	Write Characteristic Descriptors	Write Request Write Response Error Response
		Write Long Characteristic Descriptors	Prepare Write Request Prepare Write Response Prepare Write Request Prepare Write Response Error Response

1577

Annex B
(Informative)
Supporting Atomic Measurement Operation in BLE

B.1 Atomic Measurement Resource Type in OCF

Most OCF healthcare devices adopt the Atomic Measurement feature. Atomic Measurement Resource Type is a specialisation of a Collection to ensure that the Client can only access the Properties of the linked Resources as a single group. Thus, if an OCF device corresponding to a BLE device implements Atomic Measurement Resource Type, the BLE Bridging Function should guarantee that BLE GATT Characteristic values corresponding to properties of the Atomic Measurement Resource Type can be retrieved in atomic way.

B.2 Case 1. One Characteristic covers all properties of an Atomic Measurement Resource Type

In OCF-BLE mapping, a Service can be mapped to multiple OCF Resources and “a Characteristic” in a Service can be mapped to Properties in multiple OCF Resources. In general, “Value of a Characteristic” is a byte stream (see Figure B-1, byte stream is a value of “blood pressure measurement Characteristic”). Usually “value of a Characteristic” includes multiple fields like below example and each field can be mapped to a property of OCF Resource.

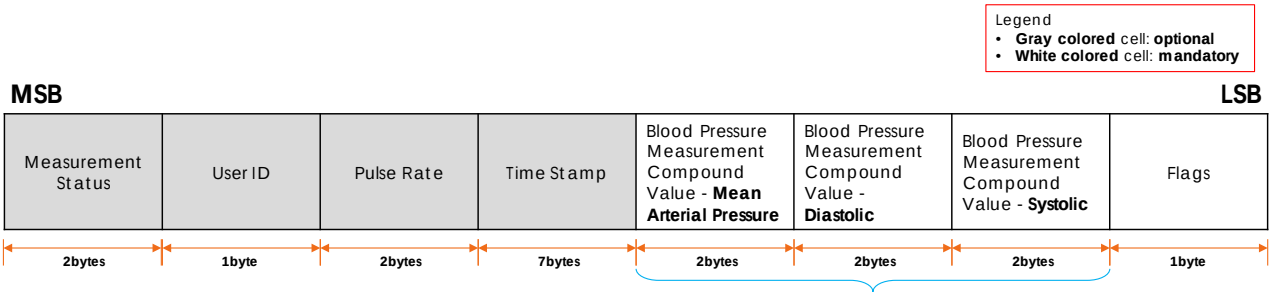


Figure B-1 – Value of blood pressure measurement Characteristic

For blood pressure device, “blood pressure measurement Characteristic” can cover all properties in bloodpressuremonitor-am. So if BLE GATT client (OCF-BLE Bridge Platform) uses “Read Characteristic Value” operation, it can get all values corresponding to all properties in bloodpressuremonitor-am at one time (atomic operation); see Figure B-2 for an example flow.

Figure B-2 – Read characteristic value example

B.3 Case 2. Multiple Characteristics cover all properties of an Atomic Measurement Resource Type

For glucose meter, 2 Characteristics (glucose measurement Characteristic, glucose measurement context Characteristic) cover all properties in glucosemeter-am. In this case, a BLE GATT client (OCF-BLE Bridge Platform) can use “Read Multiple Characteristic Values” operation to get multiple Characteristic values at one time; please see Figure B-3 for an example flow.

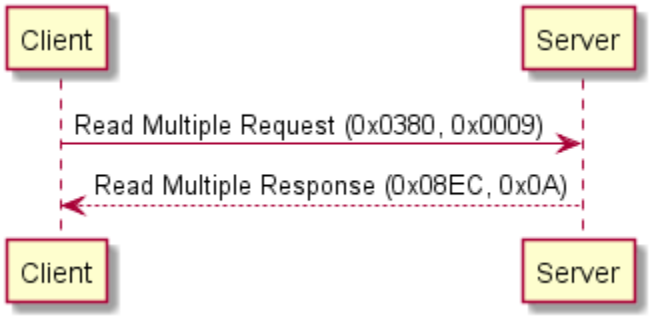


Figure B-3 – Read multiple characteristics value example

However, some BLE GATT server may not support all operations except for “Notification”. In this case, a Characteristic value includes “sequence number” field, so BLE GATT client (OCF-BLE Bridge Platform) can make a set of values which are measured at the same time by using it.

Figure B-4 and Figure B-5 are two Characteristics of glucose Service.

MSB LSB

Sensor Status Annunciation	Sample Location	Type	Glucose Concentration (kg/L or mol/L)	Time Offset	Base Time	Sequence Number	Flags
2 bytes	4 bits	4 bits	2 bytes	2 bytes	7 bytes	2 bytes	1 byte

Figure B-4 – Value of glucose measurement Characteristic

MSB LSB

HbA1c	Medication (kg or L)	Medication ID	Exercise Intensity	Exercise Duration	Health	Tester	Meal	Carbohydrate (kg)	Carbohydrate ID	Extended Flags	Sequence Number	Flags
2 bytes	2 bytes	1 byte	1 byte	2 bytes	4 bits	4 bits	1 byte	2 bytes	1 byte	1 byte	2 bytes	1 byte

Figure B-5 – Value of glucose measurement context Characteristic

