

OCF Resource to BLE Mapping

VERSION 2.2.5 | January 2022



OPEN CONNECTIVITY
FOUNDATION™

CONTACT admin@openconnectivity.org
Copyright Open Connectivity Foundation, Inc. © 2022.
All Rights Reserved.



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.

Copyright © 2019-2022 Open Connectivity Foundation, Inc. All rights reserved.

Copying or other form of reproduction and/or distribution of these works are strictly prohibited

CONTENTS

Introduction.....	vi
1 Scope.....	1
2 Normative references	1
3 Terms, definitions, symbols and abbreviated terms.....	2
3.1 Terms and definitions.....	2
3.2 Symbols and abbreviated terms	2
4 Document conventions and organization.....	2
4.1 Conventions.....	2
4.2 Notation	2
5 Theory of operation	3
5.1 Interworking approach	3
5.2 Mapping syntax.....	3
6 BLE translation.....	4
6.1 Operational scenarios	4
6.1.1 Introduction	4
6.1.2 Use case for BLE bridging	4
6.2 Requirements specific to BLE bridging function.....	5
6.2.1 General	5
6.2.2 Requirements specific to BLE	5
6.2.3 Exposing BLE GATT servers to OCF clients	5
7 Device type mapping	16
7.1 Introduction.....	16
7.2 BLE Profile to OCF device types	16
8 BLE profile to resource equivalence	17
8.1 Introduction.....	17
8.2 BLE services to OCF resources	17
9 Detailed mappings.....	18
9.1 Introduction.....	18
9.2 Blood pressure mapping	18
9.2.1 Derived model	18
9.2.2 Property definition	18
9.2.3 Derived model definition	20
9.3 Glucose measurement mapping	23
9.3.1 Derived model	23
9.3.2 Property definition	23
9.3.3 Derived model definition	27
9.4 Health thermometer mapping	32
9.4.1 Derived model	32
9.4.2 Property definition	33
9.4.3 Derived model definition	34
9.5 Weight scale mapping.....	35
9.5.1 Derived model	35

9.5.2	Property definition	35
9.5.3	Derived model definition	38
Annex A (Informative)	BLE GATT based data model	42
A.1	BLE GATT based data model & GATT features	42
A.1.1	Introduction	42
A.1.2	Profile dependency	42
A.1.3	Configurations and roles	42
A.1.4	GATT profile hierarchy	42
Annex B (Informative)	Supporting atomic measurement operation in BLE	46
B.1	Atomic measurement resource type in OCF	46
B.2	Case 1. One characteristic covers all properties of an atomic measurement resource type	46
B.3	Case 2. multiple characteristics cover all properties of an atomic measurement resource type	47

Figures

Figure 1 – OCF-BLE Bridge Platform Components.....	4
Figure 2 – BLE Bridging use case in real life.....	5
Figure 3 – Translation mapping rule illustration.....	6
Figure 4 – An example for 1:N mapping between BLE Characteristic and OCF Properties	7
Figure 5 – Initialization.....	13
Figure 6 – Resource Discovery	14
Figure 7 – Create Resource	14
Figure 8 – Retrieve Resource	14
Figure 9 – Update Resource	15
Figure 10 – Delete Resource	15
Figure 11 – Set Notification and send Notification.....	16
Figure A-1 – profile dependencies	42
Figure A-2 – GATT profile hierarchy.....	43
Figure B-1 – Value of blood pressure measurement Characteristic	46
Figure B-2 – Read characteristic value example	46
Figure B-3 – Read multiple characteristics value example.....	47
Figure B-4 – Value of glucose measurement Characteristic	47
Figure B-5 – Value of glucose measurement context Characteristic	47

Tables

Table 1 – Translation rule between BLE and OCF data model.....	5
Table 2 – BLE to OCF translation example (Blood Pressure Device).....	6
Table 3 – BLE GATT-based Profile – OCF Resource mapping	7
Table 4 – URI mapping example	8
Table 5 – "oic.wk.d" Resource Type definition.....	9
Table 6 – "oic.wk.p" Resource Type definition.....	11
Table 7 – "oic.wk.con.p" Resource Type definition	12
Table 8 – Protocol translation rule between BLE and OCF	13
Table 9 – BLE Profile to OCF Device Type Mapping	16
Table 10 – BLE Services to OCF Resource Type Mapping.....	17
Table 11 – The Property mapping for "org.bluetooth.characteristic.blood_pressure_measurement".	18
Table 12 – The Properties of "org.bluetooth.characteristic.blood_pressure_measurement". ...	19
Table 13 – The Property mapping for "org.bluetooth.characteristic.glucose_measurement". .	23
Table 14 – The Properties of "org.bluetooth.characteristic.glucose_measurement".	24
Table 15 – The Property mapping for "org.bluetooth.characteristic.glucose_measurement_context".	24
Table 16 – The Properties of "org.bluetooth.characteristic.glucose_measurement_context".	26
Table 17 – The Property mapping for "org.bluetooth.characteristic.temperature_measurement".....	33
Table 18 – The Properties of "org.bluetooth.characteristic.temperature_measurement".	33
Table 19 – The Property mapping for "org.bluetooth.characteristic.weight_measurement". ...	35
Table 20 – The Properties of "org.bluetooth.characteristic.weight_measurement".	36
Table 21 – The Property mapping for "org.bluetooth.characteristic.body_composition_measurement".	36
Table 22 – The Properties of "org.bluetooth.characteristic.body_composition_measurement".	37
Table A-1 – GATT Features and ATT protocol	43

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.

The OCF specification suite is made up of nineteen discrete documents, the documents fall into logical groupings as described herein:

- Core framework
 - Core Specification
 - Security Specification
 - Onboarding Tool Specification
- Bridging framework and bridges
 - Bridging Specification
 - Resource to Alljoyn Interface Mapping Specification
 - OCF Resource to oneM2M Resource Mapping Specification
 - OCF Resource to BLE Mapping Specification
 - OCF Resource to EnOcean Mapping Specification
 - OCF Resource to LWM2M Mapping Specification
 - OCF Resource to UPlus Mapping Specification
 - OCF Resource to Zigbee Cluster Mapping Specification
 - OCF Resource to Z-Wave Mapping Specification
- Resource and Device models
 - Resource Type Specification
 - Device Specification
- Core framework extensions
 - Easy Setup Specification
 - Core Optional Specification
- OCF Cloud
 - Cloud API for Cloud Services Specification

- Device to Cloud Services Specification
- Cloud Security Specification

OCF Resource to BLE Mapping Specification

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40

Scope

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

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>

41 **Terms, definitions, symbols and abbreviated terms**

42 **3.1 Terms and definitions**

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

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

- 47 – ISO Online browsing platform: available at <https://www.iso.org/obp>
- 48 – IEC Electropedia: available at <http://www.electropedia.org/>

49 **3.1.1**

50 **GATT-based profile**

51 BLE profile using procedures and operating models provided by GATT profile

52 **3.2 Symbols and abbreviated terms**

53	ATT	Attribute protocol
54	GAP	Generic Access Profile
55	GATT	Generic Attribute profile

56 **Document conventions and organization**

57 **4.1 Conventions**

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

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

65 **4.2 Notation**

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

68 Required (or shall or mandatory).

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

72 Recommended (or should).

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

80 Allowed (or allowed).

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

84 Conditionally allowed (CA)

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

87 Conditionally required (CR)

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

91 DEPRECATED

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

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

99 Words that are emphasized are printed in *italic*.

100 Theory of operation

101 5.1 Interworking approach

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

105 5.2 Mapping syntax

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

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

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

```
113 "<BLE Service Name>" : {  
114     "type": "object",  
115     "properties": {  
116         "<a value field in BLE Characteristic value>" : {  
117             "x-ocf-conversion" : {  
118                 "x-ocf-alias": "<corresponding OCF Resource type>",  
119                 "x-to-ocf": [  
120                     ...  
121                     ...  
122                 ],  
123                 "x-from-ocf": [  
124                     "N/A"  
125                 ]  
126             }  
127         }  
128     }
```

- 129 }
 130 – <BLE Service Name>: this is fully qualified name of a BLE Service (e.g.
 131 "org.bluetooth.characteristic.blood_pressure_measurement")
 132 – <a value field in BLE Characteristic value>: a Characteristic value is byte stream which is
 133 composed of multiple value fields. "A value field in BLE Characteristic value" is a description
 134 for one of them.
 135 – <corresponding OCF Resource type>: an OCF Resource type which is corresponding to this
 136 BLE Service.
 137 – "N/A": in BLE Bridging, most of the BLE devices are read only. So there is no specific value to
 138 be written to the BLE devices from OCF Devices. Therefore, nothing is described in "x-from-
 139 ocf" translation clause. "N/A" is used to describe this case.

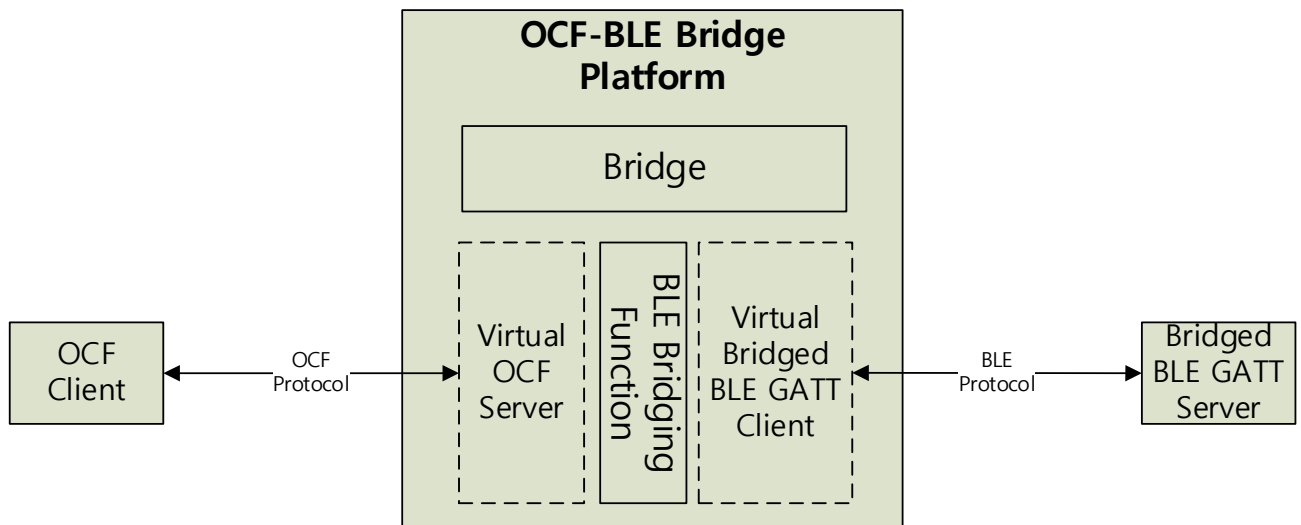
140 **BLE translation**

141 **6.1 Operational scenarios**

142 **6.1.1 Introduction**

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

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

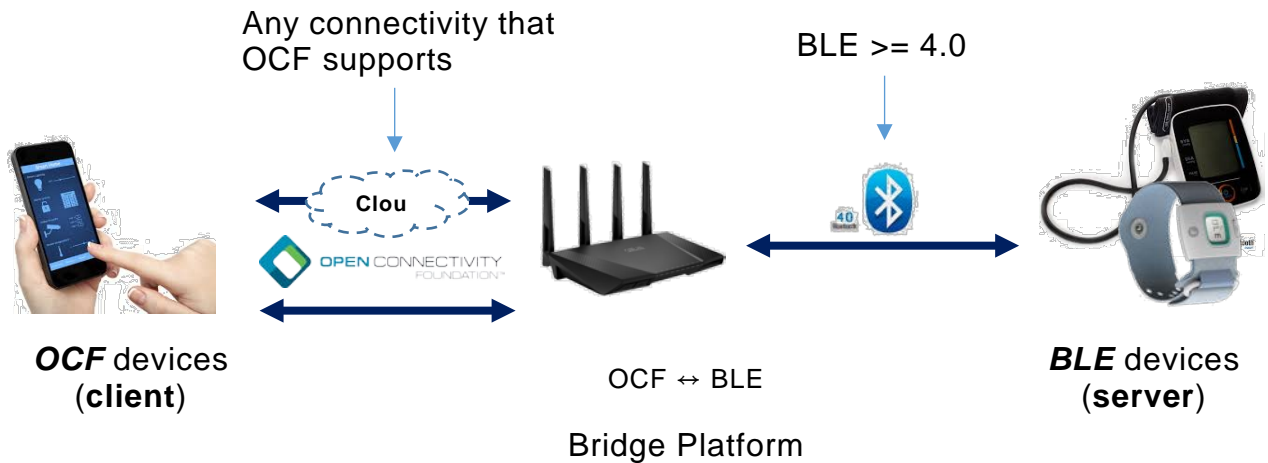


149

150 **Figure 1 – OCF-BLE Bridge Platform Components**

151 **6.1.2 Use case for BLE bridging**

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



157

158

Figure 2 – BLE Bridging use case in real life

159 **6.2 Requirements specific to BLE bridging function**

160 **6.2.1 General**

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

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

166 **6.2.2 Requirements specific to BLE**

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

169 **6.2.3 Exposing BLE GATT servers to OCF clients**

170 **6.2.3.1 General**

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

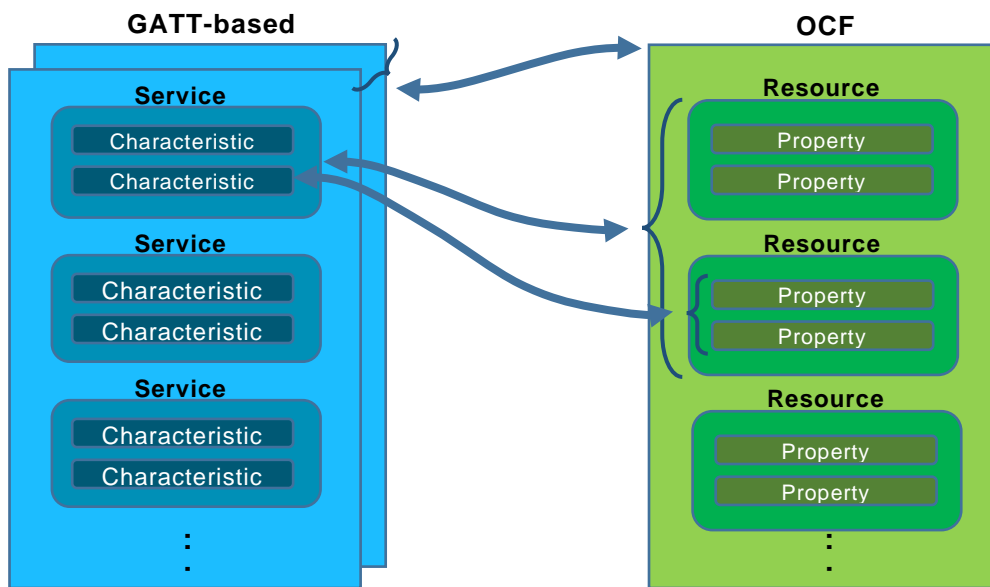
173 Basic translation rule between BLE Service/Characteristic model and OCF Resource model is
 174 described in Table 1.

175 **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

176

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



185
186

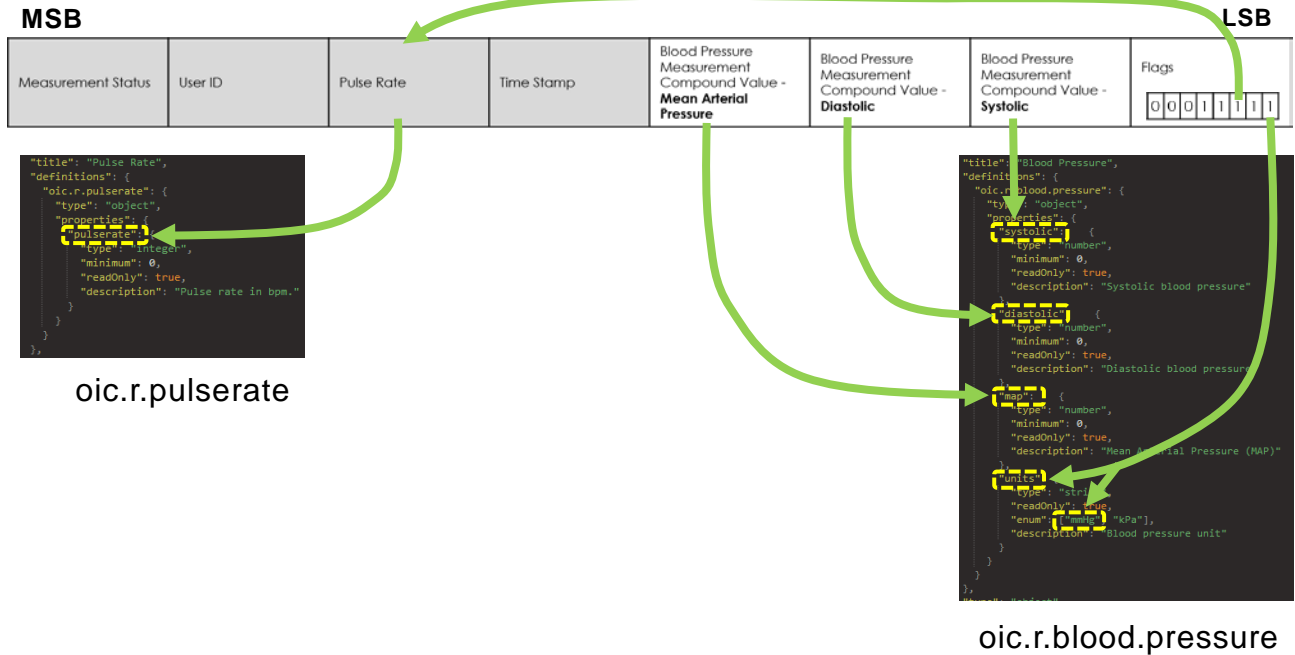
Figure 3 – Translation mapping rule illustration

187

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"

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



191

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

193 **6.2.3.2 Translation for well-defined set**

194 **6.2.3.2.1 General**

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

197 **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"	
	Device Information Service		"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.ffmpeg"	
Device Information Service			"oic.wk.d"	
			"oic.wk.p"	

198 **6.2.3.2.2 URI for virtual OCF resource**

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

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

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

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

- 206 – URI for Collection Resource: */<BLE Service name without prefix "org.bluetooth.service">* (e.g.
207 BLE Service "Health Thermometer (org.bluetooth.service.health_thermometer)":
208 /health_thermometer)
- 209 – URI for each OCF Resource link: */<OCF Resource Type value of corresponding linked
210 Resource without prefix "oic.r">* (e.g. /temperature for "oic.r.temperature",
211 /body.location.temperature for "oic.r.body.location.temperature")

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

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

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

216 **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")

217

218 **6.2.3.2.3 Common properties of resource type**

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

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

223 **6.2.3.2.4 Platform resource ("rt" of "oic.wk.p")**

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

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

230 **6.2.3.2.5 Device resource ("rt" of "oic.wk.d")**

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

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

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

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

240 **6.2.3.3 Exposing a BLE GATT server as a virtual OCF server**

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

245 **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

246
247
248
249

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.

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

252 **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)	"mnmli"	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	"mnsi"	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

253
254
255
256
257
258

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.	

259
260
261

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.

262 **6.2.3.4 On-the-fly translation**

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

265 **6.2.3.5 Protocol translation between BLE and OCF**

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

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

277 **Table 8 – Protocol translation rule between BLE and OCF**

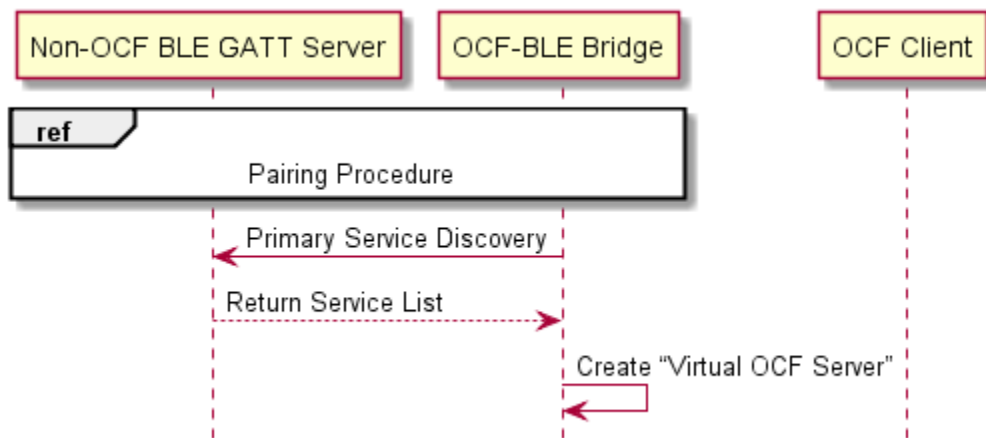
BLE GATT Feature	OCF CRUDN
N/A (Not Supported)	CREATE
Characteristic Value Read	RETRIEVE
Characteristic Value Write	UPDATE
N/A (Not Supported)	DELETE
Characteristic Descriptor Value Write	NOTIFY request
Characteristic Value Notification/Indication	NOTIFICATION response

278

279 **6.2.3.6 Illustrative OCF to BLE translation flows**

280 **6.2.3.6.1 Initialization**

281 Figure 5 shows the initial pairing procedure.

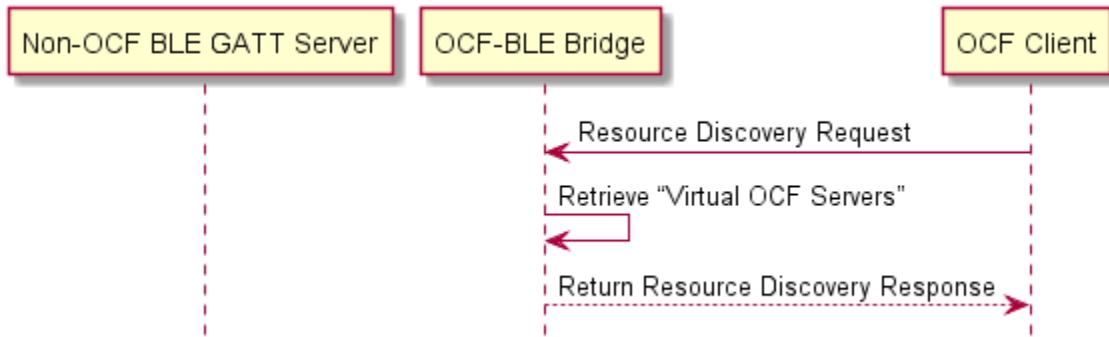


282

283 **Figure 5 – Initialization**

284 **6.2.3.6.2 Resource discovery**

285 Figure 6 shows the resource discovery procedure.

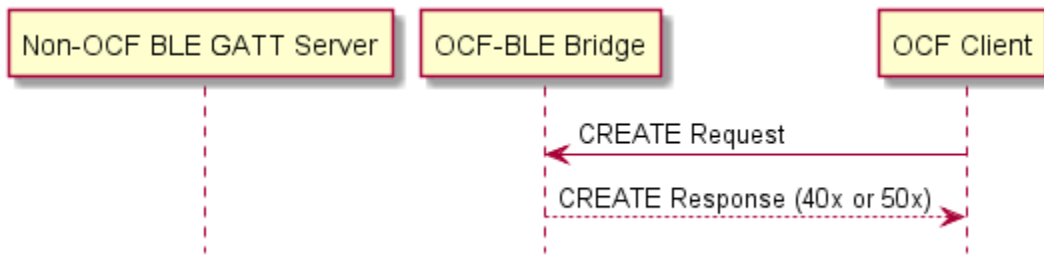


286

287 **Figure 6 – Resource Discovery**

288 **6.2.3.6.3 Create resource**

289 Figure 7 illustrates Resource creation.

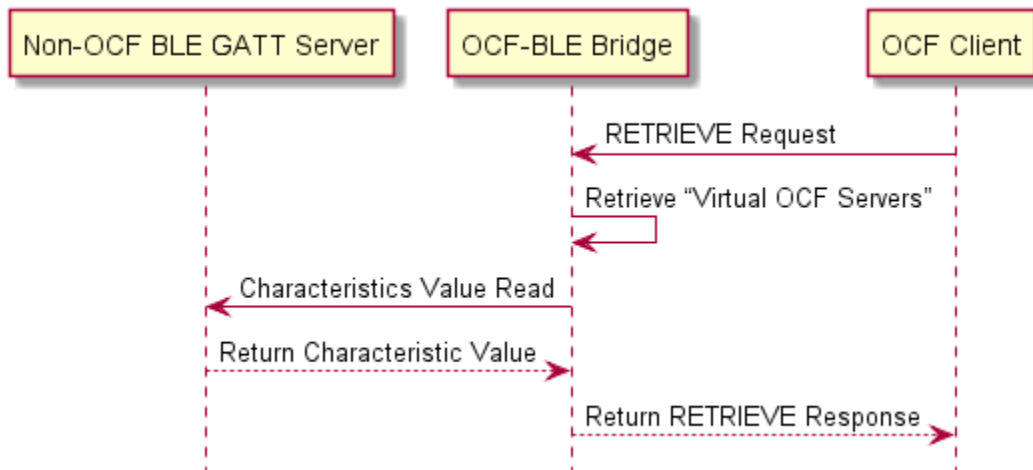


290

291 **Figure 7 – Create Resource**

292 **6.2.3.6.4 Retrieve resource**

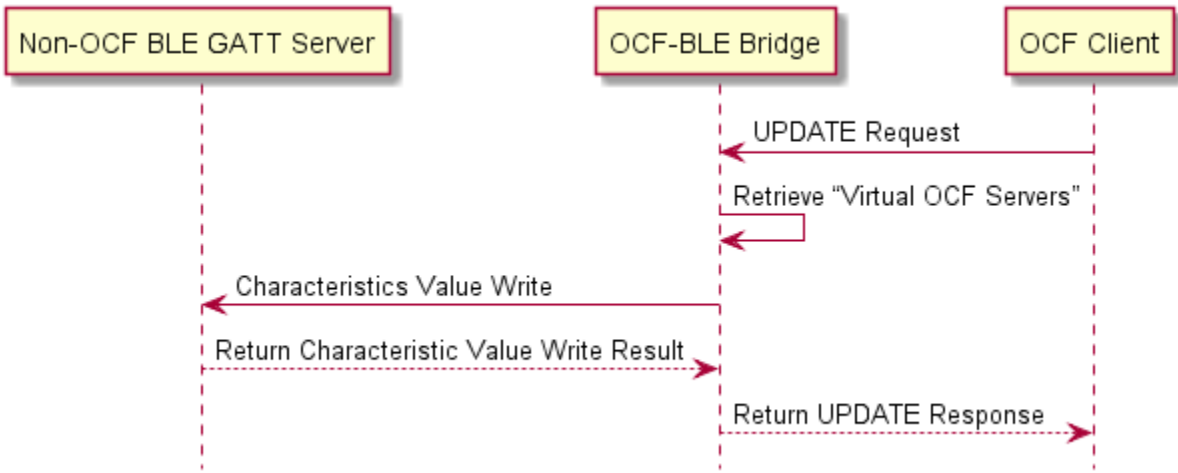
293 Figure 8 illustrates Resource RETRIEVAL.



294

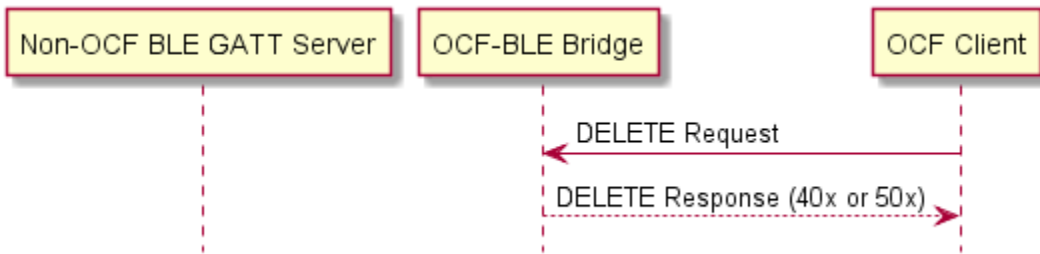
295 **Figure 8 – Retrieve Resource**

296 **6.2.3.6.5 Update resource**
 297 Figure 9 illustrates Resource UPDATE.



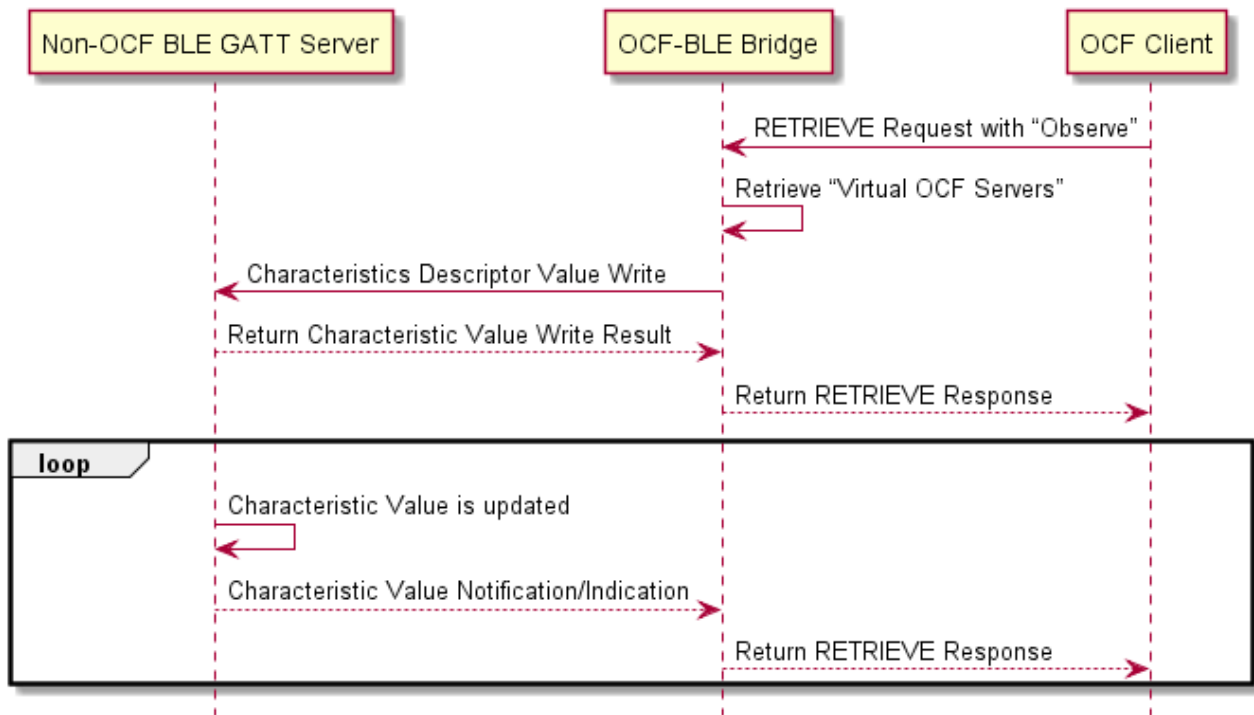
298
 299 **Figure 9 – Update Resource**

300 **6.2.3.6.6 Delete resource**
 301 Figure 10 illustrates Resource DELETE. Note that this only applies to Resources that were created.



302
 303 **Figure 10 – Delete Resource**

304 **6.2.3.6.7 Set notification & send notification**
 305 Figure 11 illustrates the establishment and sending of a notification.



306

307

Figure 11 – Set Notification and send Notification

308

309 6.2.3.7 Error handling

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

315 Device type mapping

316 7.1 Introduction

317 This clause contains the mappings to OCF Device Types.

318 7.2 BLE Profile to OCF device types

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

321

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"

322

323 **BLE profile to resource equivalence**

324 **8.1 Introduction**

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

327 **8.2 BLE services to OCF resources**

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

330

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"
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"

331

332 Detailed mappings

333 9.1 Introduction

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

337 9.2 Blood pressure mapping

338 9.2.1 Derived model

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

340 9.2.2 Property definition

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

343

344

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

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

347 **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

348 9.2.3 Derived model definition

```

349 {
350   "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.BLP.json#",
351   "$schema": "http://json-schema.org/draft-04/schema#",
352   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
353 reserved.",
354   "title": "Blood Pressure",
355   "definitions": {
356     "byte": {
357       "type": "integer",
358       "minimum": 0,
359       "maximum": 255
360     },
361     "byteArray": {
362       "type": "array",
363       "items": { "$ref": "#/definitions/byte" },
364       "minItems": 1,
365       "uniqueItems": false
366     },
367     "org.bluetooth.characteristic.blood_pressure_measurement" : {
368       "type" : "object",
369       "properties": {
370         "blood_pressure_measurement[length - 3 : length - 1]": {
371           "$ref": "#/definitions/byteArray",
372           "description": "Blood Pressure Measurement Compound Value -
373 Systolic",
374           "x-ocf-conversion": {
375             "x-ocf-alias": "oic.r.blood.pressure",
376             "x-to-ocf": [
377               "def
378 ieee11073_Sfloat_2_Float(sfloat_value):",
379               "# reserved value for Infinity or NaN
380 (Not a Number)",
381               "reserved_float_values = {",
382                 "0x07FE:math.inf, # +INFINITY",
383                 "0x07FF:math.nan, # NaN (Not a
384 Number)",
385                 "0x0800:math.nan, # NRes (Not at this
386 Resolution)",
387                 "0x0801:math.nan, # Reserved for
388 future",
389                 "0x0802:-math.inf # -INFINITY",
390               "}",
391               "mantissa = sfloat_value & 0x0FFF",
392               "exponent = sfloat_value >> 12",
393               "if (exponent >= 0x0008):",
394                 "exponent = -((0x000F + 1) -
395 exponent)",
396               "output = 0",
397               "if (mantissa >= 0x07FE and mantissa <=
398 0x0802):",
399                 "output =
400 reserved_float_values[mantissa]",
401               "else:",
402                 "if (mantissa >= 0x0800):",
403                   "mantissa = -((0x0FFF + 1) -
404 mantissa)",
405                   "magnitude = pow(10.0, exponent)",

```

```

407         "output = (mantissa * magnitude)",
408         "return output",
409         "length = len(blood_pressure_measurement)",
410         "flags = blood_pressure_measurement[length -
411 1]",
412         "oic.r.blood.pressure.systolic =
413 ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 3 : length - 1])",
414         "oic.r.blood.pressure.units = \"mmHg\" if
415 (flags & 0x01) else \"kPa\"",
416     ],
417     "x-from-ocf": [
418         "N/A"
419     ]
420 },
421     "blood_pressure_measurement[length - 5 : length - 3]": {
422         "$ref": "#/definitions/byteArray",
423         "description": "Blood Pressure Measurement Compound Value -
424 Diastolic",
425         "x-ocf-conversion": {
426             "x-ocf-alias": "oic.r.blood.pressure",
427             "x-to-ocf": [
428                 "def
429 ieee11073_Sfloat_2_Float(sfloat_value):",
430                 "# reserved value for Infinity or NaN
431 (Not a Number)",
432                 "reserved_float_values = {",
433                 "0x07FE:math.inf, # +INFINITY",
434                 "0x07FF:math.nan, # NaN (Not a
435 Number)",
436                 "0x0800:math.nan, # NRes (Not at this
437 Resolution)",
438                 "0x0801:math.nan, # Reserved for
439 future",
440                 "0x0802:-math.inf # -INFINITY",
441                 "}",
442                 "mantissa = sfloat_value & 0x0FFF",
443                 "exponent = sfloat_value >> 12",
444                 "if (exponent >= 0x0008):",
445                 "exponent = -((0x000F + 1) -
446 exponent)",
447                 "output = 0",
448                 "if (mantissa >= 0x07FE and mantissa <=
449 0x0802):",
450                 "output =
451 reserved_float_values[mantissa]",
452                 "else:",
453                 "if (mantissa >= 0x0800):",
454                 "mantissa = -((0x0FFF + 1) -
455 mantissa)",
456                 "magnitude = pow(10.0, exponent)",
457                 "output = (mantissa * magnitude)",
458                 "return output",
459                 "length = len(blood_pressure_measurement)",
460                 "oic.r.blood.pressure.diastolic =
461 ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 5 : length - 3])"
462             ],
463             "x-from-ocf": [
464                 "N/A"
465             ]
466         }
467     },
468     "blood_pressure_measurement[length - 7 : length - 5]": {
469         "$ref": "#/definitions/byteArray",
470         "description": "Blood Pressure Measurement Compound Value -
471 Mean Arterial Pressure",
472         "x-ocf-conversion": {
473             "x-ocf-alias": "oic.r.blood.pressure",
474             "x-to-ocf": [
475                 "def
476 ieee11073_Sfloat_2_Float(sfloat_value):",

```

```

478                                     "# reserved value for Infinity or NaN
479 (Not a Number)",
480                                     "reserved_float_values = {",
481                                     "0x07FE:math.inf, # +INFINITY",
482                                     "0x07FF:math.nan, # NaN (Not a
483 Number)",
484                                     "0x0800:math.nan, # NRes (Not at this
485 Resolution)",
486                                     "0x0801:math.nan, # Reserved for
487 future",
488                                     "0x0802:-math.inf # -INFINITY",
489                                     "}",
490                                     "mantissa = sfloat_value & 0x0FFF",
491                                     "exponent = sfloat_value >> 12",
492                                     "if (exponent >= 0x0008):",
493                                     "exponent = -((0x000F + 1) -
494 exponent)",
495                                     "output = 0",
496                                     "if (mantissa >= 0x07FE and mantissa <=
497 0x0802):",
498                                     "output =
499 reserved_float_values[mantissa]",
500                                     "else:",
501                                     "if (mantissa >= 0x0800):",
502                                     "mantissa = -((0x0FFF + 1) -
503 mantissa)",
504                                     "magnitude = pow(10.0, exponent)",
505                                     "output = (mantissa * magnitude)",
506                                     "return output",
507                                     "length = len(blood_pressure_measurement)",
508                                     "oic.r.blood.pressure.map =
509 ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 7 : length - 5])"
510                                     ],
511                                     "x-from-ocf": [
512                                     "N/A"
513                                     ]
514                                     },
515                                     },
516                                     "blood_pressure_measurement[length - 7 - timestamp_len - 2 : length
517 - 7 - timestamp_len]": {
518                                     "$ref": "#/definitions/byteArray",
519                                     "description": "Pulse Rate",
520                                     "x-ocf-conversion": {
521                                     "x-ocf-alias": "oic.r.pulserate",
522                                     "x-to-ocf": [
523                                     "def
524 ieee11073_Sfloat_2_Float(sfloat_value):",
525                                     "# reserved value for Infinity or NaN
526 (Not a Number)",
527                                     "reserved_float_values = {",
528                                     "0x07FE:math.inf, # +INFINITY",
529                                     "0x07FF:math.nan, # NaN (Not a
530 Number)",
531                                     "0x0800:math.nan, # NRes (Not at this
532 Resolution)",
533                                     "0x0801:math.nan, # Reserved for
534 future",
535                                     "0x0802:-math.inf # -INFINITY",
536                                     "}",
537                                     "mantissa = sfloat_value & 0x0FFF",
538                                     "exponent = sfloat_value >> 12",
539                                     "if (exponent >= 0x0008):",
540                                     "exponent = -((0x000F + 1) -
541 exponent)",
542                                     "output = 0",
543                                     "if (mantissa >= 0x07FE and mantissa <=
544 0x0802):",
545                                     "output =
546 reserved_float_values[mantissa]",
547                                     "else:",
548                                     "if (mantissa >= 0x0800):",

```

```

549                                     "mantissa = -((0x0FFF + 1) -
550 mantissa)",
551                                     "magnitude = pow(10.0, exponent)",
552                                     "output = (mantissa * magnitude)",
553                                     "return output",
554                                     "length = len(blood_pressure_measurement)",
555                                     "flags = blood_pressure_measurement[length -
556 1]",
557                                     "timestamp_len = 7 if (flags & 0x02) else 0",
558                                     "oic.r.pulserate.pulserate =
559 ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 7 - timestamp_len - 2 : length - 7 -
560 timestamp_len])"
561                                     ],
562                                     "x-from-ocf": [
563                                         "N/A"
564                                     ]
565                                 }
566                             }
567                         }
568                     }
569                 },
570             "type": "object",
571             "allOf": [
572                 { "$ref": "#/definitions/byte" },
573                 { "$ref": "#/definitions/byteArray" },
574                 { "$ref": "#/definitions/org.bluetooth.characteristic.blood_pressure_measurement" }
575             ],
576             "required": [
577                 "blood_pressure_measurement[length - 3 : length - 1]",
578                 "blood_pressure_measurement[length - 5 : length - 3]"
579             ]
580         }
581     }
582 }
583 }
584 }

```

585 9.3 Glucose measurement mapping

586 9.3.1 Derived model

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

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

589 9.3.2 Property definition

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

592 **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

		<pre> 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" </pre>	
glucose_measurement[length - 1 - 2 - timeoffset_len - 10]	oic.r.glucose.samplelocation	<pre> 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] </pre>	N/A

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

595 **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

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

598 **Table 15 – The Property mapping for**
599 **"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	<pre> 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): </pre>	N/A

		oic.r.glucose.carb.carb = ieee11073_Sfloat_2_Float(glucose_measurement_c ontext[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_c ontext[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

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

602
603

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

604 9.3.3 Derived model definition

```

605 {
606   "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.GLP.json#",
607   "$schema": "http://json-schema.org/draft-04/schema#",
608   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
609 reserved.",
610   "title": "Glucose",
611   "definitions": {
612     "byte": {
613       "type": "integer",
614       "minimum": 0,
615       "maximum": 255
616     },
617     "byteArray": {
618       "type": "array",
619       "items": { "$ref": "#/definitions/byte" },
620       "minItems": 1,
621       "uniqueItems": false
622     },
623     "org.bluetooth.characteristic.glucose_measurement": {
624       "type": "object",
625       "properties": {
626         "glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len -
627 10]": {
628           "$ref": "#/definitions/byteArray",
629           "description": "Glucose Concentration",
630           "x-ocf-conversion": {
631             "x-ocf-alias": "oic.r.glucose",
632             "x-to-ocf": [
633               "def ieee11073_Sfloat_2_Float(sfloat_value):",
634               "# reserved value for Infinity or NaN (Not a Number)",
635               "reserved_float_values = {",
636                 "0x07FE:math.inf, # +INFINITY",
637                 "0x07FF:math.nan, # NaN (Not a Number)",
638                 "0x0800:math.nan, # NRes (Not at this Resolution)",
639

```

```

640         "0x0801:math.nan, # Reserved for future",
641         "0x0802:-math.inf # -INFINITY",
642     }",
643     "mantissa = sfloat_value & 0x0FFF",
644     "exponent = sfloat_value >> 12",
645     "if (exponent >= 0x0008):",
646         "exponent = -((0x000F + 1) - exponent)",
647     "output = 0",
648     "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
649         "output = reserved_float_values[mantissa]",
650     "else:",
651         "if (mantissa >= 0x0800):",
652             "mantissa = -((0x0FFF + 1) - mantissa)",
653             "magnitude = pow(10.0, exponent)",
654             "output = (mantissa * magnitude)",
655         "return output",
656     "length = len(glucose_measurement)",
657     "flags = glucose_measurement[length - 1]",
658     "timeoffset_len = 2 if (flags & 0x01) else 0",
659     "if (flags & 0x02) == True:",
660         " glucose = ieee11073_Sfloat_2_Float(glucose_measurement[length - 2 -
661 timeoffset_len - 10 : length - timeoffset_len - 10])",
662         " oic.r.glucose.glucose = (glucose * 1000) if (flags & 0x04) else
663 (glucose * 0.1 * 1000 * 1000)",
664         " oic.r.glucose.units = 'mmol/L' if (flags & 0x04) else 'mg/dL'",
665     "if (flags & 0x02) == False:",
666         " oic.r.glucose.glucose = 0",
667         " oic.r.glucose.units = 'mmol/L'"
668     ],
669     "x-from-ocf": [
670         "N/A"
671     ]
672 }
673 },
674 "glucose_measurement[length - 1 - 2 - timeoffset_len - 10]": {
675     "$ref": "#/definitions/byteArray",
676     "description": "Sample Location",
677     "x-ocf-conversion": {
678         "x-ocf-alias": "oic.r.glucose.samplelocation",
679         "x-to-ocf": [
680             "length = len(glucose_measurement)",
681             "flags = glucose_measurement[length - 1]",
682             "timeoffset_len = 2 if (flags & 0x01) else 0",
683             "if (flags & 0x02):",
684                 " samplelocation = { 1:'finger', 2:'ast', 3:'earlobe',
685 4:'ctrlsolution' }",
686                 " oic.r.glucose.samplelocation.samplelocation =
687 samplelocation[glucose_measurement[length - 1 - 2 - timeoffset_len - 10] & 0xf0]"
688             ],
689             "x-from-ocf": [
690                 "N/A"
691             ]
692         }
693     }
694 },
695 },
696
697     "org.bluetooth.characteristic.glucose_measurement_context": {
698         "type": "object",
699     "properties": {
700         "glucose_measurement_context[length - carb_len - extflags_len - 3 : length - 1 -
701 extflags_len - 3]": {
702             "$ref": "#/definitions/byteArray",
703             "description": "Carbohydrate",
704             "x-ocf-conversion": {
705                 "x-ocf-alias": "oic.r.glucose.carb",
706                 "x-to-ocf": [
707                     "def ieee11073_Sfloat_2_Float(sfloat_value):",
708                         "# reserved value for Infinity or NaN (Not a Number)",
709                         "reserved_float_values = {",
710                             "0x07FE:math.inf, # +INFINITY",

```

```

711         "0x07FF:math.nan, # NaN (Not a Number)",
712         "0x0800:math.nan, # NRes (Not at this Resolution)",
713         "0x0801:math.nan, # Reserved for future",
714         "0x0802:-math.inf # -INFINITY",
715     },
716     "mantissa = sfloat_value & 0x0FFF",
717     "exponent = sfloat_value >> 12",
718     "if (exponent >= 0x0008):",
719         "exponent = -((0x000F + 1) - exponent)",
720     "output = 0",
721     "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
722         "output = reserved_float_values[mantissa]",
723     "else:",
724         "if (mantissa >= 0x0800):",
725             "mantissa = -((0x0FFF + 1) - mantissa)",
726             "magnitude = pow(10.0, exponent)",
727             "output = (mantissa * magnitude)",
728         "return output",
729     "length = len(glucose_measurement_context)",
730     "flags = glucose_measurement_context[length - 1]",
731     "extflags_len = 1 if (flags & 0x80) else 0",
732     "carb_len = 3 if (flags & 0x01) else 0",
733     "if (flags & 0x01): ",
734     "    oic.r.glucose.carb.carb =
735 ieee11073_Sfloat_2_Float(glucose_measurement_context[length - carb_len - extflags_len - 3 : length
736 - 1 - extflags_len - 3]) * 1000"
737     ],
738     "x-from-ocf": [
739         "N/A"
740     ]
741     }
742 },
743 "glucose_measurement_context[length - 1 - extflags_len - 3]": {
744     "$ref": "#/definitions/byteArray",
745     "description": "Carbohydrate ID",
746     "x-ocf-conversion": {
747         "x-ocf-alias": "oic.r.glucose.carb",
748         "x-to-ocf": [
749             "length = len(glucose_measurement_context)",
750             "flags = glucose_measurement_context[length - 1]",
751             "extflags_len = 1 if (flags & 0x80) else 0",
752             "if (flags & 0x01): ",
753             "    meal = { 1:'breakfast', 2:'lunch', 3:'dinner', 4:'snack',
754 5:'drink', 6:'supper', 7:'brunch' }",
755             "    oic.r.glucose.carb.meal = meal[glucose_measurement_context[length -
756 1 - extflags_len - 3]]"
757         ],
758         "x-from-ocf": [
759             "N/A"
760         ]
761     }
762 },
763 "glucose_measurement_context[length - 2 - health_len - meal_len - carb_len -
764 extflags_len - 3]": {
765     "$ref": "#/definitions/byteArray",
766     "description": "Exercise Intensity",
767     "x-ocf-conversion": {
768         "x-ocf-alias": "oic.r.glucose.exercise",
769         "x-to-ocf": [
770             "length = len(glucose_measurement_context)",
771             "flags = glucose_measurement_context[length - 1]",
772             "extflags_len = 1 if (flags & 0x80) else 0",
773             "carb_len = 3 if (flags & 0x01) else 0",
774             "meal_len = 1 if (flags & 0x02) else 0",
775             "health_len = 1 if (flags & 0x04) else 0",
776             "if (flags & 0x08): ",
777             "    oic.r.glucose.exercise.exercise =
778 glucose_measurement_context[length - 2 - health_len - meal_len - carb_len - extflags_len - 3]"
779         ],
780         "x-from-ocf": [
781             "N/A"

```

```

782         ]
783     }
784 },
785     "glucose_measurement_context[length - hbalc_len - medication_len - exercise_len -
786 health_len - meal_len - carb_len - extflags_len - 3 : length - medication_len - exercise_len -
787 health_len - meal_len - carb_len - extflags_len - 3]": {
788     "$ref": "#/definitions/byteArray",
789     "description": "HbA1c",
790     "x-ocf-conversion": {
791     "x-ocf-alias": "oic.r.glucose.hbalc",
792     "x-to-ocf": [
793     "def ieee11073_Sfloat_2_Float(sfloat_value):",
794     "# reserved value for Infinity or NaN (Not a Number)",
795     "reserved_float_values = {",
796     "0x07FE:math.inf, # +INFINITY",
797     "0x07FF:math.nan, # NaN (Not a Number)",
798     "0x0800:math.nan, # NRes (Not at this Resolution)",
799     "0x0801:math.nan, # Reserved for future",
800     "0x0802:-math.inf # -INFINITY",
801     "}",
802     "mantissa = sfloat_value & 0x0FFF",
803     "exponent = sfloat_value >> 12",
804     "if (exponent >= 0x0008):",
805     "    exponent = -((0x000F + 1) - exponent)",
806     "output = 0",
807     "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
808     "    output = reserved_float_values[mantissa]",
809     "else:",
810     "    if (mantissa >= 0x0800):",
811     "        mantissa = -((0x0FFF + 1) - mantissa)",
812     "        magnitude = pow(10.0, exponent)",
813     "        output = (mantissa * magnitude)",
814     "    return output",
815     "length = len(glucose_measurement_context)",
816     "flags = glucose_measurement_context[length - 1]",
817     "extflags_len = 1 if (flags & 0x80) else 0",
818     "carb_len = 3 if (flags & 0x01) else 0",
819     "meal_len = 1 if (flags & 0x02) else 0",
820     "health_len = 1 if (flags & 0x04) else 0",
821     "exercise_len = 3 if (flags & 0x08) else 0",
822     "medication_len = 3 if (flags & 0x10) else 0",
823     "hbalc_len = 2 if (flags & 0x40) else 0",
824     "if (flags & 0x40):",
825     "    oic.r.glucose.hbalc.hbalc =
826 ieee11073_Sfloat_2_Float(glucose_measurement_context[length - hbalc_len - medication_len -
827 exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - medication_len -
828 exercise_len - health_len - meal_len - carb_len - extflags_len - 3])",
829     ],
830     "x-from-ocf": [
831     "N/A"
832     ]
833     }
834 },
835     "glucose_measurement_context[length - health_len - meal_len - carb_len -
836 extflags_len - 3]": {
837     "$ref": "#/definitions/byteArray",
838     "description": "Health, Tester",
839     "x-ocf-conversion": {
840     "x-ocf-alias": "oic.r.glucose.health",
841     "x-to-ocf": [
842     "length = len(glucose_measurement_context)",
843     "flags = glucose_measurement_context[length - 1]",
844     "extflags_len = 1 if (flags & 0x80) else 0",
845     "carb_len = 3 if (flags & 0x01) else 0",
846     "meal_len = 1 if (flags & 0x02) else 0",
847     "health_len = 1 if (flags & 0x04) else 0",
848     "if (flags & 0x04):",
849     "    health = { 1:'minor', 2:'major', 3:'menses', 4:'stress',
850 5:'none' }",
851     "    oic.r.glucose.health.health =
852 health[glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3] &

```

```

853 0xf0]",
854         "tester = { 1:'self', 2:'hcp', 3:'lab' }",
855         "oic.r.glucose.tester.tester =
856 tester[glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3] &
857 0xf0]"
858     ],
859     "x-from-ocf": [
860         "N/A"
861     ]
862 },
863 },
864 "glucose_measurement_context[length - meal_len - carb_len - extflags_len - 3]: {
865     "$ref": "#/definitions/byteArray",
866     "description": "Meal",
867     "x-ocf-conversion": {
868         "x-ocf-alias": "oic.r.glucose.meal",
869         "x-to-ocf": [
870             "length = len(glucose_measurement_context)",
871             "flags = glucose_measurement_context[length - 1]",
872             "extflags_len = 1 if (flags & 0x80) else 0",
873             "carb_len = 3 if (flags & 0x01) else 0",
874             "meal_len = 1 if (flags & 0x02) else 0",
875             "if (flags & 0x02): ",
876             "meal = { 1:'preprandial', 2:'postprandial', 3:'fasting',
877 4:'casual', 5:'bedtime' }",
878             "oic.r.glucose.meal.meal = meal[glucose_measurement_context[length -
879 meal_len - carb_len - extflags_len - 3]]"
880         ],
881         "x-from-ocf": [
882             "N/A"
883         ]
884     }
885 },
886 "glucose_measurement_context[length - medication_len - exercise_len - health_len -
887 meal_len - carb_len - extflags_len - 3 : length - 1 - exercise_len - health_len - meal_len -
888 carb_len - extflags_len - 3]: {
889     "$ref": "#/definitions/byteArray",
890     "description": "Medication",
891     "x-ocf-conversion": {
892         "x-ocf-alias": "oic.r.glucose.medication",
893         "x-to-ocf": [
894             "def ieee11073_Sfloat_2_Float(sfloat_value):",
895             "# reserved value for Infinity or NaN (Not a Number)",
896             "reserved_float_values = {",
897             "0x07FE:math.inf, # +INFINITY",
898             "0x07FF:math.nan, # NaN (Not a Number)",
899             "0x0800:math.nan, # NRes (Not at this Resolution)",
900             "0x0801:math.nan, # Reserved for future",
901             "0x0802:-math.inf # -INFINITY",
902             "}",
903             "mantissa = sfloat_value & 0x0FFF",
904             "exponent = sfloat_value >> 12",
905             "if (exponent >= 0x0008):",
906             "    exponent = -((0x000F + 1) - exponent)",
907             "output = 0",
908             "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
909             "    output = reserved_float_values[mantissa]",
910             "else:",
911             "    if (mantissa >= 0x0800):",
912             "        mantissa = -((0x0FFF + 1) - mantissa)",
913             "    magnitude = pow(10.0, exponent)",
914             "    output = (mantissa * magnitude)",
915             "return output",
916             "length = len(glucose_measurement_context)",
917             "flags = glucose_measurement_context[length - 1]",
918             "extflags_len = 1 if (flags & 0x80) else 0",
919             "carb_len = 3 if (flags & 0x01) else 0",
920             "meal_len = 1 if (flags & 0x02) else 0",
921             "health_len = 1 if (flags & 0x04) else 0",
922             "exercise_len = 3 if (flags & 0x08) else 0",
923             "medication_len = 3 if (flags & 0x10) else 0",

```

```

924             "hbalc_len = 2 if (flags & 0x40) else 0",
925             "if (flags & 0x10): ",
926             "    medication =
927 iiee11073_Sfloat_2_Float(glucose_measurement_context[length - medication_len - exercise_len -
928 health_len - meal_len - carb_len - extflags_len - 3 : length - 1 - exercise_len - health_len -
929 meal_len - carb_len - extflags_len - 3])",
930             "    oic.r.glucose.medication.medication = medication * 1000",
931             "    oic.r.glucose.medication.units = 'mL' if (flags & 0x20) else 'mg'"
932         ],
933         "x-from-ocf": [
934             "N/A"
935         ]
936     },
937 },
938     "glucose_measurement_context[length - 1 - exercise_len - health_len - meal_len -
939 carb_len - extflags_len - 3]": {
940         "$ref": "#/definitions/byteArray",
941         "description": "Medication ID",
942         "x-ocf-conversion": {
943             "x-ocf-alias": "oic.r.glucose.medication",
944             "x-to-ocf": [
945                 "length = len(glucose_measurement_context)",
946                 "flags = glucose_measurement_context[length - 1]",
947                 "extflags_len = 1 if (flags & 0x80) else 0",
948                 "carb_len = 3 if (flags & 0x01) else 0",
949                 "meal_len = 1 if (flags & 0x02) else 0",
950                 "health_len = 1 if (flags & 0x04) else 0",
951                 "exercise_len = 3 if (flags & 0x08) else 0",
952                 "if (flags & 0x10): ",
953                 "    regimen = { 1:'rapidacting', 2:'shortacting',
954 3:'intermediateacting', 4:'longacting', 5:'premix' }",
955                 "    oic.r.glucose.medication.regimen =
956 regimen[ glucose_measurement_context[length - 1 - exercise_len - health_len - meal_len - carb_len -
957 extflags_len - 3] ]"
958             ],
959             "x-from-ocf": [
960                 "N/A"
961             ]
962         }
963     }
964 },
965 },
966 },
967 },
968 "type": "object",
969 },
970 "allOf": [
971     { "$ref": "#/definitions/byte" },
972     { "$ref": "#/definitions/byteArray" },
973     { "$ref":
974 "#/definitions/org.bluetooth.characteristic.org.bluetooth.characteristic.glucose_measurement" },
975     { "$ref":
976 "#/definitions/org.bluetooth.characteristic.org.bluetooth.characteristic.glucose_measurement_context" }
977 ],
978 },
979 },
980 "required": [
981     "glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len -
982 10]"
983 ]
984 }
985

```

986 9.4 Health thermometer mapping

987 9.4.1 Derived model

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

989 **9.4.2 Property definition**

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

992 **Table 17 – The Property mapping for**
 993 **"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_measur ement[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

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

996 **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

997 9.4.3 Derived model definition

```
998 {
999   {
1000     "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.HTP.json#",
1001     "$schema": "http://json-schema.org/draft-04/schema#",
1002     "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
1003 reserved.",
1004     "title": "Health Thermometer",
1005     "definitions": {
1006       "byte": {
1007         "type": "integer",
1008         "minimum": 0,
1009         "maximum": 255
1010       },
1011       "byteArray": {
1012         "type": "array",
1013         "items": { "$ref": "#/definitions/byte" },
1014         "minItems": 1,
1015         "uniqueItems": false
1016       },
1017       "org.bluetooth.characteristic.temperature_measurement" : {
1018         "type": "object",
1019         "properties" : {
1020           "temperature_measurement[ length - 5 : length - 1 ]" : {
1021             "$ref": "#/definitions/byteArray",
1022             "description": "Temperature",
1023             "x-ocf-conversion": {
1024               "x-ocf-alias": "oic.r.temperature",
1025               "x-to-ocf": [
1026                 "# convert IEEE11073 FLOAT to float",
1027                 "def ieee11073_Float_2_Float(float_value):",
1028                 "# reserved value for Infinity or NaN
1029 (Not a Number)",
1030                 "reserved_float_values = {",
1031                 "  0x007FFFFE:math.inf, # +INFINITY",
1032                 "  0x007FFFFFF:math.nan, # NaN (Not a
1033 Number)",
1034                 "  0x00800000:math.nan, # NRes (Not at
1035 this Resolution)",
1036                 "  0x00800001:math.nan, # Reserved for
1037 future",
1038                 "  0x00800002:-math.inf # -INFINITY",
1039                 "}",
1040                 "mantissa = float_value & 0x00FFFFFF",
1041                 "exponent = float_value >> 24",
1042                 "if (exponent >= 0x00000080):",
1043                 "  exponent = -((0x000000FF + 1) -
1044 exponent)",
1045                 "output = 0",
1046                 "if (mantissa >= 0x007FFFFE and mantissa
1047 <= 0x00800002):",
1048                 "  output =
1049 reserved_float_values[mantissa]",
1050                 "else:",
1051                 "  if (mantissa >= 0x00800000):",
1052                 "    mantissa = -((0x00FFFFFF + 1) -
1053 mantissa)",
1054                 "  magnitude = pow(10.0, exponent)",
1055                 "  output = (mantissa * magnitude)",
1056                 "  return output",
1057                 "length = len(temperature_measurement)",
1058                 "flags = temperature_measurement[length -
1059 1]",
1060                 "oic.r.temperature.temperature =
1061 ieee11073_Float_2_Float(temperature_measurement[ length - 5 : length - 1 ])",
1062                 "oic.r.temperature.units = 'F' if (flags &
1063 0x01) else 'C'
1064               ],
1065               "x-from-ocf": [
1066                 "N/A"
```

```

1067         ]
1068     }
1069 },
1070     "temperature_measurement[ length - temperaturetype_len -
1071 timestamp_len - 5 ]" : {
1072         "$ref": "#/definitions/byteArray",
1073         "description": "Temperature Type",
1074         "x-ocf-conversion": {
1075             "x-ocf-alias": "oic.r.body.location.temperature",
1076             "x-to-ocf": [
1077                 "length = len(temperature_measurement)",
1078                 "flags = temperature_measurement[length -
1079 1]",
1080                 "timestamp_len = 7 if (flags & 0x02) 0",
1081                 "temperaturetype_len = 1 if (flags & 0x04)
1082 0",
1083                 "if (flags & 0x04):",
1084                 "    bloc = { 1:'xxx', 2:'body', 3:'ear',
1085 4:'finger', 5:'gastro', 6:'mouth', 7:'rectum', 8:'toe', 9:'tympaanum' }",
1086                 "    oic.r.body.location.temperature.bloc =
1087 bloc[temperature_measurement[ length - temperaturetype_len - timestamp_len - 5 ] ]"
1088             ],
1089             "x-from-ocf": [
1090                 "N/A"
1091             ]
1092         }
1093     }
1094 }
1095 }
1096 },
1097 "type": "object",
1098 "allOf": [
1099     { "$ref": "#/definitions/byte" },
1100     { "$ref": "#/definitions/byteArray" },
1101     { "$ref": "#/definitions/org.bluetooth.characteristic.temperature_measurement" }
1102 ],
1103 "required": [
1104     "temperature_measurement[ length - 5 : length - 1 ]"
1105 ]
1106 }
1107 }
1108 }
1109 }
1110 }

```

1111 9.5 Weight scale mapping

1112 9.5.1 Derived model

1113 The derived model: "org.bluetooth.characteristic.weight_measurement".

1114 The derived model: "org.bluetooth.characteristic.body_composition_measurement".

1115 9.5.2 Property definition

1116 Table 19 provides the detailed per Property mapping for
1117 "org.bluetooth.characteristic.weight_measurement".

1118 **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

1119 Table 20 provides the details of the Properties that are part of
1120 "org.bluetooth.characteristic.weight_measurement".

1121 **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

1122 Table 21 provides the detailed per Property mapping for
1123 "org.bluetooth.characteristic.body_composition_measurement".

1124 **Table 21 – The Property mapping for**
1125 **"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.wat er	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

		<pre> Offm_len = 2 if (flags_lowerbyte & 0x40) else Oslm_len = 2 if (flags_lowerbyte & 0x80) else Obwm_len = 2 if (flags_upperbyte & 0x01) else 0if (flags_lowerbyte & 0x01): oic.r.body.water.bwater = int.from_bytes(body_composition_measurement[len gth - 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' </pre>	
<pre> body_composition_measurement[le ngth - 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] </pre>	oic.r.body.slm	<pre> length = len(body_composition_measurement)flags_upperbyt e = body_composition_measurement[length - 2]flags_lowerbyte = body_composition_measurement[length - 1]timestamp_len = 7 if (flags_lowerbyte & 0x02) else Ouserid_len = 1 if (flags_lowerbyte & 0x04) else Obasal_len = 2 if (flags_lowerbyte & 0x08) else Omuscle_len = 2 if (flags_lowerbyte & 0x10) else Omm_len = 2 if (flags_lowerbyte & 0x20) else Offm_len = 2 if (flags_lowerbyte & 0x40) else Oslm_len = 2 if (flags_lowerbyte & 0x80) else 0if (flags_lowerbyte & 0x01): oic.r.body.slm.bwater = int.from_bytes(body_composition_measurement[len gth - 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' </pre>	N/A
<pre> body_composition_measurement[le ngth - 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] </pre>	oic.r.body.ffm	<pre> length = len(body_composition_measurement)flags_upperbyt e = body_composition_measurement[length - 2]flags_lowerbyte = body_composition_measurement[length - 1]timestamp_len = 7 if (flags_lowerbyte & 0x02) else Ouserid_len = 1 if (flags_lowerbyte & 0x04) else Obasal_len = 2 if (flags_lowerbyte & 0x08) else Omuscle_len = 2 if (flags_lowerbyte & 0x10) else Omm_len = 2 if (flags_lowerbyte & 0x20) else Offm_len = 2 if (flags_lowerbyte & 0x40) else 0if (flags_lowerbyte & 0x01): oic.r.body.ffm.bwater = int.from_bytes(body_composition_measurement[len gth - 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' </pre>	N/A

1126 Table 22 provides the details of the Properties that are part of
1127 "org.bluetooth.characteristic.body_composition_measurement".

1128 **Table 22 – The Properties of**
1129 **"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

1130 **9.5.3 Derived model definition**

```

1131 {
1132   "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.WSS.json#",
1133   "$schema": "http://json-schema.org/draft-04/schema#",
1134   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
1135 reserved.",
1136   "title": "Weight Scale",
1137   "definitions": {
1138     "byte": {
1139       "type": "integer",
1140       "minimum": 0,
1141       "maximum": 255
1142     },
1143     "byteArray": {
1144       "type": "array",
1145       "items": { "$ref": "#/definitions/byte" },
1146       "minItems": 1,
1147       "uniqueItems": false
1148     },
1149     "org.bluetooth.characteristic.weight_measurement" : {
1150       "type": "object",
1151       "properties": {
1152         "weight_measurement[length - 3 : length - 1]" : {
1153           "$ref": "#/definitions/byteArray",
1154           "description": "Weight",
1155           "x-ocf-conversion": {
1156             "x-ocf-alias": "oic.r.weight",
1157             "x-to-ocf": [
1158               "length = len(weight_measurement)",
1159               "flags = weight_measurement[length - 1]",
1160               "timeoffset_len = 7 if (flags & 0x02) else
1161 0",
1162               "oic.r.weight.weight =
1163 int.from_bytes(weight_measurement[length - 3 : length - 1], 'big')",
1164               "oic.r.weight.units = 'lb' if (flags & 0x01)
1165 else 'kg'"
1166             ],
1167             "x-from-ocf": [
1168               "N/A"
1169             ]
1170           }
1171         },
1172         "weight_measurement[length - 2 - userid_len - timeoffset_len - 3 :
1173 length - userid_len - timeoffset_len - 3]" : {
1174           "$ref": "#/definitions/byteArray",
1175           "description": "BMI",
1176           "x-ocf-conversion": {
1177             "x-ocf-alias": "oic.r.bmi",
1178             "x-to-ocf": [
1179               "length = len(weight_measurement)",
1180               "flags = weight_measurement[length - 1]",
1181               "timeoffset_len = 7 if (flags & 0x02) else
1182 0",

```

```

1184                                     "userid_len = 1 if (flags & 0x04) else 0",
1185                                     "if (flags & 0x08):",
1186                                     "    oic.r.bmi.bmi =
1187 int.from_bytes(weight_measurement[length - 2 - userid_len - timeoffset_len - 3 : length -
1188 userid_len - timeoffset_len - 3], 'big')"
1189                                     ],
1190                                     "x-from-ocf": [
1191                                         "N/A"
1192                                     ]
1193                                 }
1194                             },
1195                             "weight_measurement[length - height_len - userid_len -
1196 timeoffset_len - 3 : length - 2 - userid_len - timeoffset_len - 3]" : {
1197                                 "$ref": "#/definitions/byteArray",
1198                                 "description": "Height",
1199                                 "x-ocf-conversion": {
1200                                     "x-ocf-alias": "oic.r.height",
1201                                     "x-to-ocf": [
1202                                         "length = len(weight_measurement)",
1203                                         "flags = weight_measurement[length - 1]",
1204                                         "timeoffset_len = 7 if (flags & 0x02) else
1205 0",
1206                                         "userid_len = 1 if (flags & 0x04) else 0",
1207                                         "height_len = 4 if (flags & 0x08) else 0",
1208                                         "if (flags & 0x08):",
1209                                         "    oic.r.height.height =
1210 int.from_bytes(weight_measurement[length - height_len - userid_len - timeoffset_len - 3 : length -
1211 2 - userid_len - timeoffset_len - 3], 'big')",
1212                                         "    oic.r.height.units = 'in' if (flags &
1213 0x01) else 'm'"
1214                                     ],
1215                                     "x-from-ocf": [
1216                                         "N/A"
1217                                     ]
1218                                 }
1219                             }
1220                         }
1221                     },
1222                 "org.bluetooth.characteristic.body_composition_measurement" : {
1223                     "type": "object",
1224                     "properties": {
1225                         "body_composition_measurement[ length - 4 : length - 2]" : {
1226                             "$ref": "#/definitions/byteArray",
1227                             "description": "Body Fat Percentage",
1228                             "x-ocf-conversion": {
1229                                 "x-ocf-alias": "oic.r.body.fat",
1230                                 "x-to-ocf": [
1231                                     "length = len(body_composition_measurement)",
1232                                     "oic.r.body.fat.bodyfat =
1233 int.from_bytes(body_composition_measurement[ length - 4 : length - 2], 'big')",
1234                                     "oic.r.body.fat.units = '%'"
1235                                 ],
1236                                 "x-from-ocf": [
1237                                     "N/A"
1238                                 ]
1239                             }
1240                         }
1241                     },
1242                     "body_composition_measurement[ length - bwm_len - slm_len - ffm_len
1243 - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len - ffm_len -
1244 mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 ]" : {
1245                         "$ref": "#/definitions/byteArray",
1246                         "description": "Body Water Mass",
1247                         "x-ocf-conversion": {
1248                             "x-ocf-alias": "oic.r.body.water",
1249                             "x-to-ocf": [
1250                                 "length = len(body_composition_measurement)",
1251                                 "flags_upperbyte =
1252 body_composition_measurement[length - 2]",
1253                                 "flags_lowerbyte =
1254 body_composition_measurement[length - 1]"

```

```

1255                                     "timestamp_len = 7 if (flags_lowerbyte &
1256 0x02) else 0",
1257                                     "userid_len = 1 if (flags_lowerbyte & 0x04)
1258 else 0",
1259                                     "basal_len = 2 if (flags_lowerbyte & 0x08)
1260 else 0",
1261                                     "muscle_len = 2 if (flags_lowerbyte & 0x10)
1262 else 0",
1263                                     "mm_len = 2 if (flags_lowerbyte & 0x20) else
1264 0",
1265                                     "ffm_len = 2 if (flags_lowerbyte & 0x40) else
1266 0",
1267                                     "slm_len = 2 if (flags_lowerbyte & 0x80) else
1268 0",
1269                                     "bwm_len = 2 if (flags_upperbyte & 0x01) else
1270 0",
1271                                     "if (flags_lowerbyte & 0x01): ",
1272                                     "    oic.r.body.water.bwater =
1273 int.from_bytes(body_composition_measurement[ length - bwm_len - slm_len - ffm_len - mm_len -
1274 muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len - ffm_len - mm_len -
1275 muscle_len - basal_len - userid_len - timestamp_len - 4 ], 'big')",
1276                                     "    oic.r.body.water.units = 'lb' if
1277 (flags_lowerbyte & 0x01) 'kg'"
1278                                     ],
1279                                     "x-from-ocf": [
1280                                         "N/A"
1281                                     ]
1282                                 },
1283                                 },
1284                                 "body_composition_measurement[ length - slm_len - ffm_len - mm_len -
1285 muscle_len - basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len - muscle_len -
1286 basal_len - userid_len - timestamp_len - 4 ]" : {
1287                                     "$ref": "#/definitions/byteArray",
1288                                     "description": "Soft Lean Mass",
1289                                     "x-ocf-conversion": {
1290                                         "x-ocf-alias": "oic.r.body.slm",
1291                                         "x-to-ocf": [
1292                                             "length = len(body_composition_measurement)",
1293                                             "flags_upperbyte =
1294 body_composition_measurement[length - 2]",
1295                                             "flags_lowerbyte =
1296 body_composition_measurement[length - 1]",
1297                                             "timestamp_len = 7 if (flags_lowerbyte &
1298 0x02) else 0",
1299                                             "userid_len = 1 if (flags_lowerbyte & 0x04)
1300 else 0",
1301                                             "basal_len = 2 if (flags_lowerbyte & 0x08)
1302 else 0",
1303                                             "muscle_len = 2 if (flags_lowerbyte & 0x10)
1304 else 0",
1305                                             "mm_len = 2 if (flags_lowerbyte & 0x20) else
1306 0",
1307                                             "ffm_len = 2 if (flags_lowerbyte & 0x40) else
1308 0",
1309                                             "slm_len = 2 if (flags_lowerbyte & 0x80) else
1310 0",
1311                                             "if (flags_lowerbyte & 0x01): ",
1312                                             "    oic.r.body.slm.bwater =
1313 int.from_bytes(body_composition_measurement[ length - slm_len - ffm_len - mm_len - muscle_len -
1314 basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len - muscle_len - basal_len -
1315 userid_len - timestamp_len - 4 ], 'big')",
1316                                             "    oic.r.body.slm.units = 'lb' if
1317 (flags_lowerbyte & 0x01) 'kg'"
1318                                             ],
1319                                             "x-from-ocf": [
1320                                                 "N/A"
1321                                             ]
1322                                         ]
1323                                     },
1324                                 "body_composition_measurement[ length - ffm_len - mm_len -
1325 muscle_len - basal_len - userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len

```



```

1326 - userid_len - timestamp_len - 4 ] : {
1327     "$ref": "#/definitions/byteArray",
1328     "description": "Fat Free Mass",
1329     "x-ocf-conversion": {
1330         "x-ocf-alias": "oic.r.body.ffm",
1331         "x-to-ocf": [
1332             "length = len(body_composition_measurement)",
1333             "flags_upperbyte =
1334 body_composition_measurement[length - 2]",
1335             "flags_lowerbyte =
1336 body_composition_measurement[length - 1]",
1337             "timestamp_len = 7 if (flags_lowerbyte &
1338 0x02) else 0",
1339             "userid_len = 1 if (flags_lowerbyte & 0x04)
1340 else 0",
1341             "basal_len = 2 if (flags_lowerbyte & 0x08)
1342 else 0",
1343             "muscle_len = 2 if (flags_lowerbyte & 0x10)
1344 else 0",
1345             "mm_len = 2 if (flags_lowerbyte & 0x20) else
1346 0",
1347             "ffm_len = 2 if (flags_lowerbyte & 0x40) else
1348 0",
1349             "if (flags_lowerbyte & 0x01): ",
1350             "    oic.r.body.ffm.bwater =
1351 int.from_bytes(body_composition_measurement[ length - ffm_len - mm_len - muscle_len - basal_len -
1352 userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len - userid_len -
1353 timestamp_len - 4 ], 'big')",
1354             "    oic.r.body.ffm.units = 'lb' if
1355 (flags_lowerbyte & 0x01) 'kg'"
1356         ],
1357         "x-from-ocf": [
1358             "N/A"
1359         ]
1360     }
1361 }
1362 }
1363 }
1364 },
1365
1366 "type": "object",
1367
1368 "allOf": [
1369     { "$ref": "#/definitions/byte" },
1370     { "$ref": "#/definitions/byteArray" },
1371     { "$ref": "#/definitions/org.bluetooth.characteristic.weight_measurement" },
1372     { "$ref": "#/definitions/org.bluetooth.characteristic.body_composition_measurement" }
1373 ],
1374
1375 "required": [
1376     "weight_measurement[length - 3 : length - 1]"
1377 ]
1378 }
1379

```

1380
1381
1382

Annex A (Informative) BLE GATT based data model

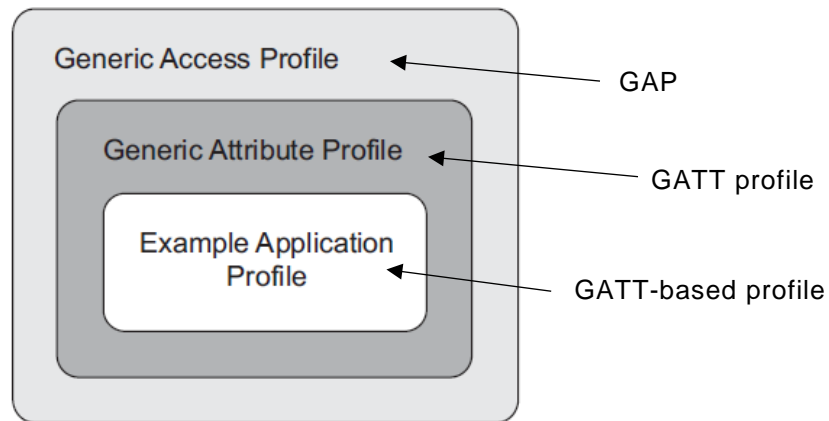
1383 A.1 BLE GATT based data model & GATT features

1384 A.1.1 Introduction

1385 The Generic Attribute Profile (GATT) defines a service framework using the Attribute Protocol.
1386 This framework defines procedures and formats of services and their characteristics. The
1387 procedures defined include discovering, reading, writing, notifying and indicating characteristics,
1388 as well as configuring the broadcast of characteristics.

1389 A.1.2 Profile dependency

1390 Figure A-1 depicts the structure and the dependencies of the profiles. A profile is dependent upon
1391 another profile if it re-uses parts of that profile by implicitly or explicitly referencing it.



1392
1393

Figure A-1 – profile dependencies

1394 A.1.3 Configurations and roles

1395 There are two roles defined in GATT profile:

- 1396 • Client: This is the device that initiates commands and requests towards the server and can
1397 receive responses, indications and notifications sent by the server.
- 1398 • Server: This is the device that accepts incoming commands and requests from the client
1399 and sends responses, indications and notifications to a client.

1400 A device can act in both roles at the same time.

1401 A.1.4 GATT profile hierarchy

1402 A.1.4.1 Introduction

1403 The GATT Profile specifies the structure in which profile data is exchanged. This structure defines
1404 basic elements such as services and characteristics, used in a profile. All of the elements are
1405 contained by Attributes. Attributes used in the ATT are containers that carry this profile data.

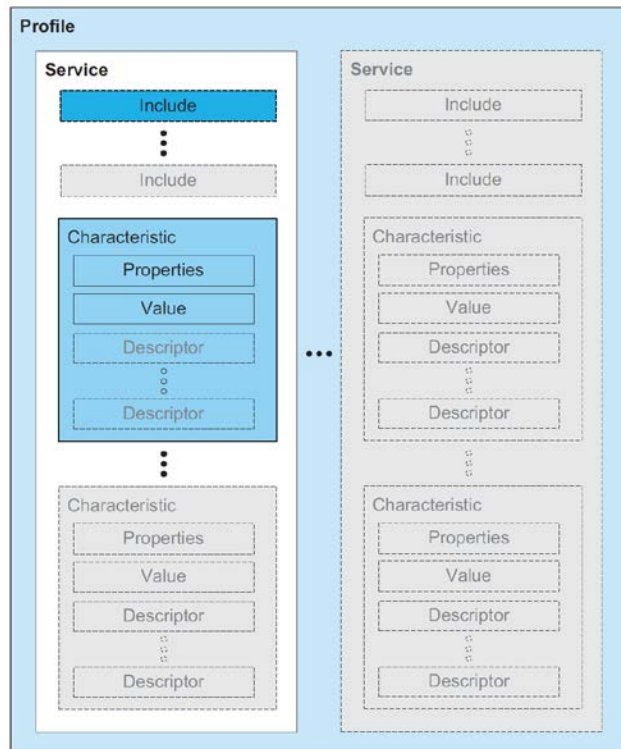
1406 The top level of the hierarchy is a profile. A profile is composed of one or more services necessary
1407 to fulfil a use case. A service is composed of characteristics or references to other services. Each
1408 characteristic contains a value and may contain optional information about the value. The service

1409 and characteristic and the components of the characteristic (i.e. value and descriptors) contain the
 1410 profile data and are all stored in Attributes on the server.

1411 Under GATT profile, entity that provides Service-Characteristics data model plays “server” role
 1412 and entity that gets data from GATT server plays “client” role.

1413 There are other application profiles based on GATT profile. They are called “GATT-based profiles”.

1414 Figure A-2 Illustrates the GATT profile hierarchy.



1415

1416

Figure A-2 – GATT profile hierarchy

1417 **A.1.4.2 Characteristic**

1418 A characteristic is a value used in a service along with properties and configuration information
 1419 about how the value is accessed and information about how the value is displayed or represented.
 1420 In GATT, a characteristic is defined by its characteristic definition. A characteristic definition
 1421 contains a characteristic declaration, characteristic properties, and a value and may contain
 1422 descriptors that describe the value or permit configuration of the server with respect to the
 1423 characteristic.

1424 **A.1.4.3 GATT features**

1425 GATT profile also supports GATT features. GATT feature defines how GATT-based data
 1426 exchanges take place. Each feature is mapped to one or more sub-procedures. These sub-
 1427 procedures describe how the ATT is used to accomplish the corresponding feature, please see
 1428 Table A-1.

1429

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

1430

1431
1432
1433

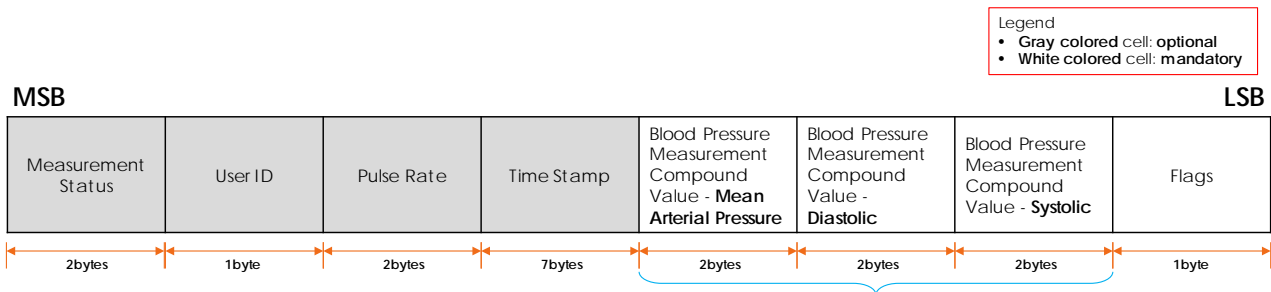
Annex B (Informative) Supporting atomic measurement operation in BLE

1434 B.1 Atomic measurement resource type in OCF

1435 Most OCF healthcare devices adopt the Atomic Measurement feature. Atomic Measurement
1436 Resource Type is a specialisation of a Collection to ensure that the Client can only access the
1437 Properties of the linked Resources as a single group. Thus, if an OCF device corresponding to a
1438 BLE device implements Atomic Measurement Resource Type, the BLE Bridging Function should
1439 guarantee that BLE GATT Characteristic values corresponding to properties of the Atomic
1440 Measurement Resource Type can be retrieved in atomic way.

1441 B.2 Case 1. One characteristic covers all properties of an atomic measurement 1442 resource type

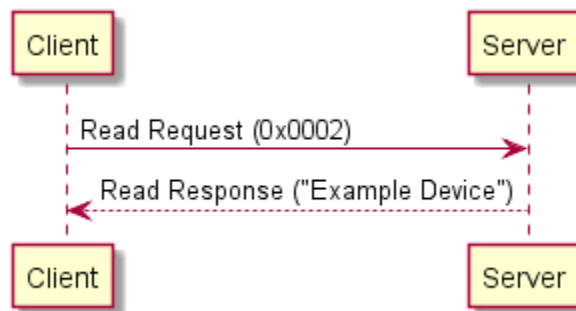
1443 In OCF-BLE mapping, a Service can be mapped to multiple OCF Resources and “a Characteristic”
1444 in a Service can be mapped to Properties in multiple OCF Resources. In general, “Value of a
1445 Characteristic” is a byte stream (see Figure B-1, byte stream is a value of “blood pressure
1446 measurement Characteristic”). Usually “value of a Characteristic” includes multiple fields like below
1447 example and each field can be mapped to a property of OCF Resource.



1448
1449

Figure B-1 – Value of blood pressure measurement Characteristic

1450 For blood pressure device, “blood pressure measurement Characteristic” can cover all properties
1451 in bloodpressuremonitor-am. So if BLE GATT client (OCF-BLE Bridge Platform) uses “Read
1452 Characteristic Value” operation, it can get all values corresponding to all properties in
1453 bloodpressuremonitor-am at one time (atomic operation); see Figure B-2 for an example flow.

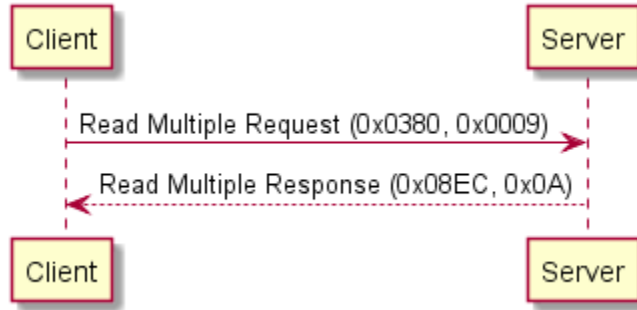


1454
1455
1456

Figure B-2 – Read characteristic value example

1457 **B.3 Case 2. multiple characteristics cover all properties of an atomic measurement**
 1458 **resource type**

1459 For glucose meter, 2 Characteristics (glucose measurement Characteristic, glucose measurement
 1460 context Characteristic) cover all properties in glucosemeter-am. In this case, a BLE GATT client
 1461 (OCF-BLE Bridge Platform) can use “Read Multiple Characteristic Values” operation to get multiple
 1462 Characteristic values at one time; please see Figure B-3 for an example flow.



1463
 1464 **Figure B-3 – Read multiple characteristics value example**

1465 However, some BLE GATT server may not support all operations except for “Notification”. In this
 1466 case, a Characteristic value includes “sequence number” field, so BLE GATT client (OCF-BLE
 1467 Bridge Platform) can make a set of values which are measured at the same time by using it.

1468 Figure B-4 and Figure B-5 are two Characteristics of glucose Service.

MSB						LSB	
Sensor Status Amunciation	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

1469
 1470 **Figure B-4 – Value of glucose measurement Characteristic**

MSB											LSB	
HbA1c	Medication (kilograms or liter)	Medication ID	Exercise Intensity	Exercise Duration	Health	Tester	Meal	Carbohydrate (kilograms)	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 bytes	1 bytes	2 bytes	1 byte

1471
 1472 **Figure B-5 – Value of glucose measurement context Characteristic**