

OCF Resource to BLE Mapping Specification

VERSION 2.1.1 | February 2020



OPEN CONNECTIVITY
FOUNDATION™

CONTACT admin@openconnectivity.org

Copyright Open Connectivity Foundation, Inc. © 2020.
All Rights Reserved.

3 Legal Disclaimer

4

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

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

18 Copyright © 2019-2020 Open Connectivity Foundation, Inc. All rights reserved.

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

20

CONTENTS

22	1	Scope	1
23	2	Normative references	1
24	3	Terms, definitions, symbols and abbreviated terms	1
25	3.1	Terms and definitions	1
26	3.2	Abbreviated terms	2
27	4	Document conventions and organization	2
28	4.1	Conventions	2
29	4.2	Notation.....	2
30	5	Theory of Operation	3
31	5.1	Interworking Approach.....	3
32	5.2	Mapping Syntax.....	3
33	6	BLE Translation	4
34	6.1	Operational Scenarios	4
35	6.1.1	Introduction	4
36	6.1.2	Use case for BLE Bridging	5
37	6.2	Requirements specific to BLE Bridging Function.....	5
38	6.2.1	General.....	5
39	6.2.2	Requirements specific to BLE.....	5
40	6.2.3	Exposing BLE GATT Servers to OCF Clients.....	5
41	7	Device Type Mapping.....	16
42	7.1	Introduction	16
43	7.2	BLE Profile to OCF Device Types	16
44	8	BLE Profile to Resource Equivalence	17
45	8.1	Introduction	17
46	8.2	BLE Services to OCF Resources	17
47	9	Detailed Mappings	18
48	9.1	Introduction	18
49	9.2	Blood Pressure Mapping.....	18
50	9.2.1	Derived model	18
51	9.2.2	Property definition	18
52	9.2.3	Derived model definition.....	20
53	9.3	Glucose Measurement Mapping.....	23
54	9.3.1	Derived model	23
55	9.3.2	Property definition	23
56	9.3.3	Derived model definition.....	27
57	9.4	Health Thermometer Mapping.....	33
58	9.4.1	Derived model	33
59	9.4.2	Property definition	33
60	9.4.3	Derived model definition.....	34
61	9.5	Weight Scale Mapping	35
62	9.5.1	Derived model	35
63	9.5.2	Property definition	35

64	9.5.3	Derived model definition.....	38
65		Annex A (Informative) BLE GATT based Data Model	42
66	A.1	BLE GATT based data model & GATT features	42
67	A.1.1	Introduction	42
68	A.1.2	Profile dependency	42
69	A.1.3	Configurations and roles	42
70	A.1.4	GATT profile hierarchy	42
71		Annex B (Informative) Supporting Atomic Measurement Operation in BLE	46
72	B.1	Atomic Measurement Resource Type in OCF.....	46
73	B.2	Case 1. One Characteristic covers all properties of an Atomic Measurement	
74		Resource Type	46
75	B.3	Case 2. Multiple Characteristics cover all properties of an Atomic Measurement	
76		Resource Type	47
77			

Figures

79	Figure 1 – OCF-BLE Bridge Platform Components	4
80	Figure 2 – BLE Bridging use case in real life	5
81	Figure 3 – Translation mapping rule illustration	6
82	Figure 4 – An example for 1:N mapping between BLE Characteristic and OCF Properties	7
83	Figure 5 – Initialization	14
84	Figure 6 – Resource Discovery	14
85	Figure 7 – Create Resource.....	14
86	Figure 8 – Retrieve Resource	15
87	Figure 9 – Update Resource	15
88	Figure 10 – Delete Resource	15
89	Figure 11 – Set Notification and send Notification.....	16
90	Figure A-1 – profile dependencies	42
91	Figure A-2 – GATT profile hierarchy	43
92	Figure B-1 – Value of blood pressure measurement Characteristic	46
93	Figure B-2 – Read characteristic value example	46
94	Figure B-3 – Read multiple characteristics value example	47
95	Figure B-4 – Value of glucose measurement Characteristic	47
96	Figure B-5 – Value of glucose measurement context Characteristic.....	47
97		

Tables

99	Table 1 – Translation rule between BLE and OCF data model	5
100	Table 2 – BLE to OCF translation example (Blood Pressure Device).....	6
101	Table 3 – BLE GATT-based Profile – OCF Resource mapping.....	7
102	Table 4 – URI mapping example.....	9
103	Table 5 – "oic.wk.d" Resource Type definition	10
104	Table 6 – "oic.wk.p" Resource Type definition	11
105	Table 7 – "oic.wk.con.p" Resource Type definition.....	13
106	Table 8 – Protocol translation rule between BLE and OCF.....	13
107	Table 9 – BLE Profile to OCF Device Type Mapping.....	16
108	Table 10 – BLE Services to OCF Resource Type Mapping	17
109	Table 11 – The Property mapping for	
110	"org.bluetooth.characteristic.blood_pressure_measurement".....	18
111	Table 12 – The Properties of "org.bluetooth.characteristic.blood_pressure_measurement"...	19
112	Table 13 – The Property mapping for	
113	"org.bluetooth.characteristic.glucose_measurement".....	23
114	Table 14 – The Properties of "org.bluetooth.characteristic.glucose_measurement".....	24
115	Table 15 – The Property mapping for	
116	"org.bluetooth.characteristic.glucose_measurement_context".....	24
117	Table 16 – The Properties of	
118	"org.bluetooth.characteristic.glucose_measurement_context".....	27
119	Table 17 – The Property mapping for	
120	"org.bluetooth.characteristic.temperature_measurement".....	33
121	Table 18 – The Properties of "org.bluetooth.characteristic.temperature_measurement".....	33
122	Table 19 – The Property mapping for "org.bluetooth.characteristic.weight_measurement"....	35
123	Table 20 – The Properties of "org.bluetooth.characteristic.weight_measurement".....	36
124	Table 21 – The Property mapping for	
125	"org.bluetooth.characteristic.body_composition_measurement".....	36
126	Table 22 – The Properties of	
127	"org.bluetooth.characteristic.body_composition_measurement".....	37
128	Table A-1 – GATT Features and ATT protocol	43
129		

130 **1 Scope**

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

133 **2 Normative references**

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

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

140 Bluetooth Core Specification 4.0
141 <https://www.bluetooth.com/specifications/bluetooth-core-specification>

142 ISO/IEC 30118-1:2019 Information technology -- Open Connectivity Foundation (OCF)
143 Specification -- Part 1: Core specification
144 <https://www.iso.org/standard/53238.html>
145 Latest version available at: https://openconnectivity.org/specs/OCF_Core_Specification.pdf

146 ISO/IEC 30118-2:2019, Information technology – Open Connectivity Foundation (OCF)
147 Specification – Part 2: Security specification
148 <https://www.iso.org/standard/74239.html>
149 Latest version available at: https://openconnectivity.org/specs/OCF_Security_Specification.pdf

150 ISO/IEC 30118-3:2019, Information technology – Open Connectivity Foundation (OCF)
151 Specification – Part 3: Bridging specification
152 <https://www.iso.org/standard/74240.html>
153 Latest version available at: https://openconnectivity.org/specs/OCF_Bridging_Specification.pdf

154 ISO/IEC 30118-4:2019, Information technology – Open Connectivity Foundation (OCF)
155 Specification – Part 4: Resource Type specification
156 <https://www.iso.org/standard/74241.html>
157 Latest version available at:
158 https://openconnectivity.org/specs/OCF_Resource_Type_Specification.pdf

159 ISO/IEC 30118-5:2019, Information technology – Open Connectivity Foundation (OCF)
160 Specification – Part 5: Device specification
161 <https://www.iso.org/standard/79389.html>
162 Latest version available at: https://openconnectivity.org/specs/OCF_Device_Specification.pdf

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

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

168 **3 Terms, definitions, symbols and abbreviated terms**

169 **3.1 Terms and definitions**

170 For the purposes of this document, the terms and definitions given in ISO/IEC 30118-1:2019,
171 ISO/IEC 30118-2:2019, and ISO/IEC 30118-3:2019 and the following apply.

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

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

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

176 **3.1.1**

177 **GATT-based Profile**

178 a BLE profile using procedures and operating models provided by GATT profile

179 **3.2 Abbreviated terms**

180 **3.2.1**

181 **ATT**

182 ATTRIBUTE protocol

183 ATT is a protocol for discovering, reading, and writing attributes on a peer device (Vol 3, part F of
184 Bluetooth Core Specification 4.0)

185 **3.2.2**

186 **GAP**

187 Generic Access Profile

188 GAP defines the generic procedures related to discovery of Bluetooth devices (idle mode
189 procedures) and link management aspects of connecting to Bluetooth devices (connecting mode
190 procedures). It also defines procedures related to use of different security levels. In addition, this
191 profile includes common format requirements for parameters accessible on the user interface level.
192 (Vol 3, part C of Bluetooth Core Specification 4.0)

193 **3.2.3**

194 **GATT**

195 Generic ATTRIBUTE profile

196 Generic Attribute Profile describes a service framework using the Attribute Protocol for discovering
197 services, and for reading and writing characteristic values on a peer device (Vol 3 part G of
198 Bluetooth Core Specification 4.0).

199 **4 Document conventions and organization**

200 **4.1 Conventions**

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

205 **4.2 Notation**

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

208 Required (or shall or mandatory).

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

212 Recommended (or should).

213 These features add functionality supported by the Mapping Specification and should be
214 implemented. Recommended features take advantage of the capabilities the Mapping
215 Specification, usually without imposing major increase of complexity. Notice that for compliance

216 testing, if a recommended feature is implemented, it shall meet the specified requirements to
217 be in compliance with these guidelines. Some recommended features could become
218 requirements in the future. The phrase "should not" indicates behavior that is permitted but not
219 recommended.

220 Allowed (or allowed).

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

224 Conditionally allowed (CA)

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

227 Conditionally required (CR)

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

231 DEPRECATED

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

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

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

240 5 Theory of Operation

241 5.1 Interworking Approach

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

245 5.2 Mapping Syntax

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

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

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

```
253 "<BLE Service Name>" : {  
254     "type": "object",  
255     "properties": {  
256         "<a value field in BLE Characteristic value>" : {  
257             "x-ocf-conversion" : {  
258                 "x-ocf-alias": "<corresponding OCF Resource type>",  
259                 "x-to-ocf": [  
260                     ...
```

```

261         ...
262     ],
263     "x-from-ocf": [
264         "N/A"
265     ]
266 }
267 }
268 }
269 }

```

- 270 – <BLE Service Name>: this is fully qualified name of a BLE Service (e.g.
271 "org.bluetooth.characteristic.blood_pressure_measurement")
- 272 – <a value field in BLE Characteristic value>: a Characteristic value is byte stream which is
273 composed of multiple value fields. "A value field in BLE Characteristic value" is a description
274 for one of them.
- 275 – <corresponding OCF Resource type>: an OCF Resource type which is corresponding to this
276 BLE Service.
- 277 – "N/A": in BLE Bridging, most of the BLE devices are read only. So there is no specific value to
278 be written to the BLE devices from OCF Devices. Therefore, nothing is described in "x-from-
279 ocf" translation clause. "N/A" is used to describe this case.

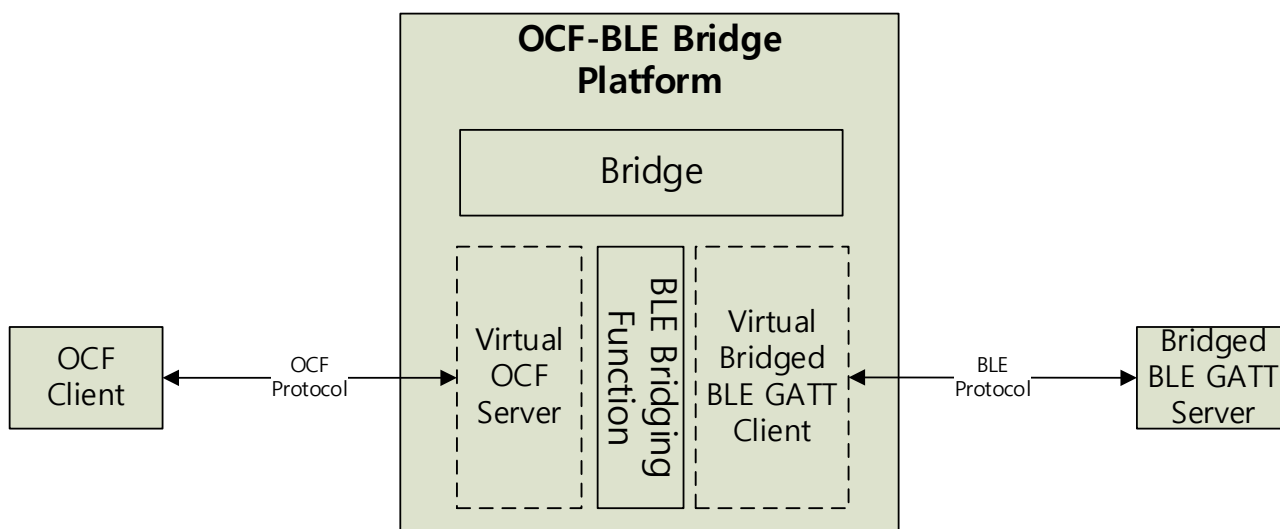
280 6 BLE Translation

281 6.1 Operational Scenarios

282 6.1.1 Introduction

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

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



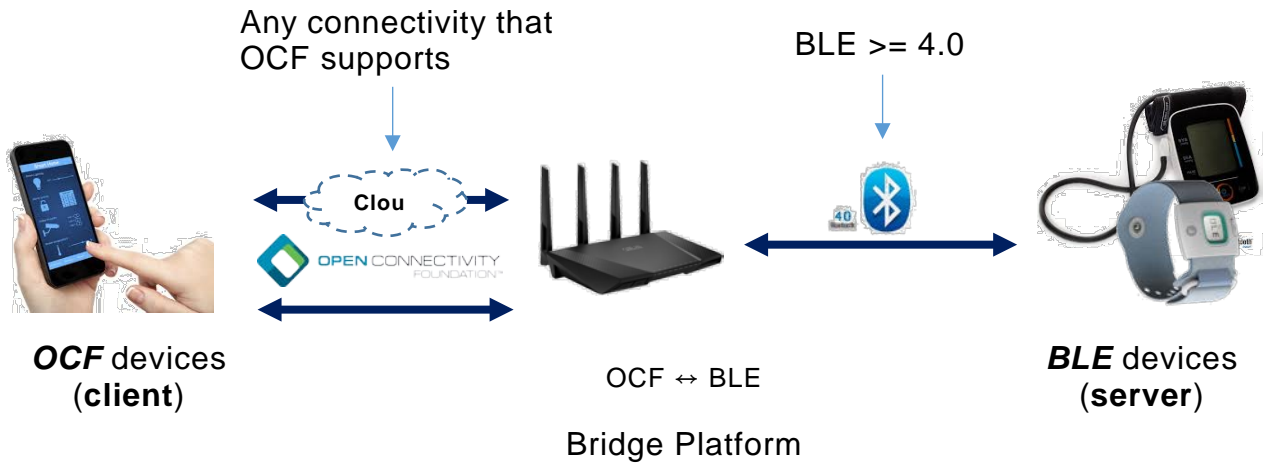
289

290

Figure 1 – OCF-BLE Bridge Platform Components

291 **6.1.2 Use case for BLE Bridging**

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



297
 298 **Figure 2 – BLE Bridging use case in real life**

299 **6.2 Requirements specific to BLE Bridging Function**

300 **6.2.1 General**

301 OCF-BLE Bridge Platform shall satisfy clause 5.2 General Requirements of ISO/IEC 30118-3:2019.

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

306 **6.2.2 Requirements specific to BLE**

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

309 **6.2.3 Exposing BLE GATT Servers to OCF Clients**

310 **6.2.3.1 General**

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

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

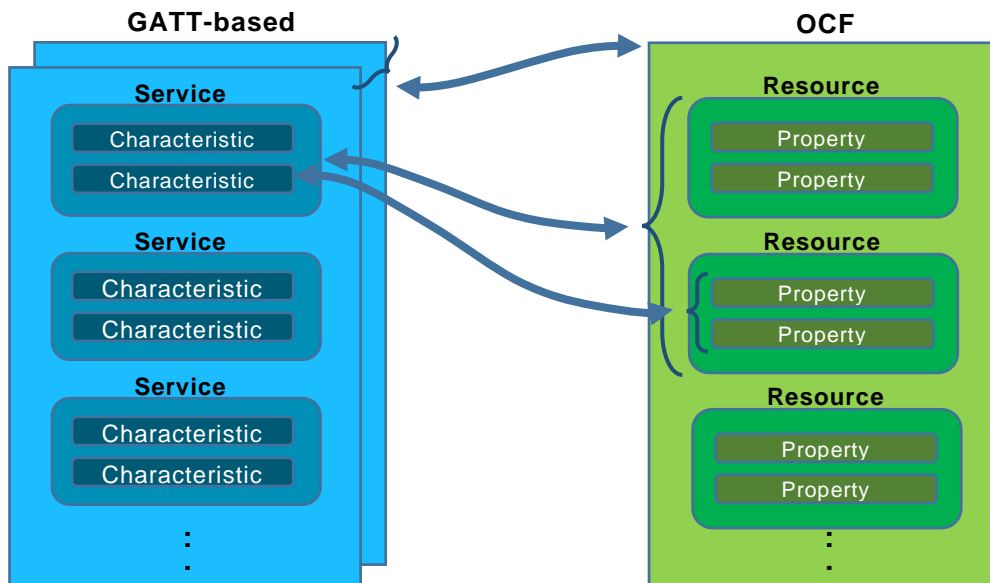
315 **Table 1 – Translation rule between BLE and OCF data model**

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

316

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



325

326

Figure 3 – Translation mapping rule illustration

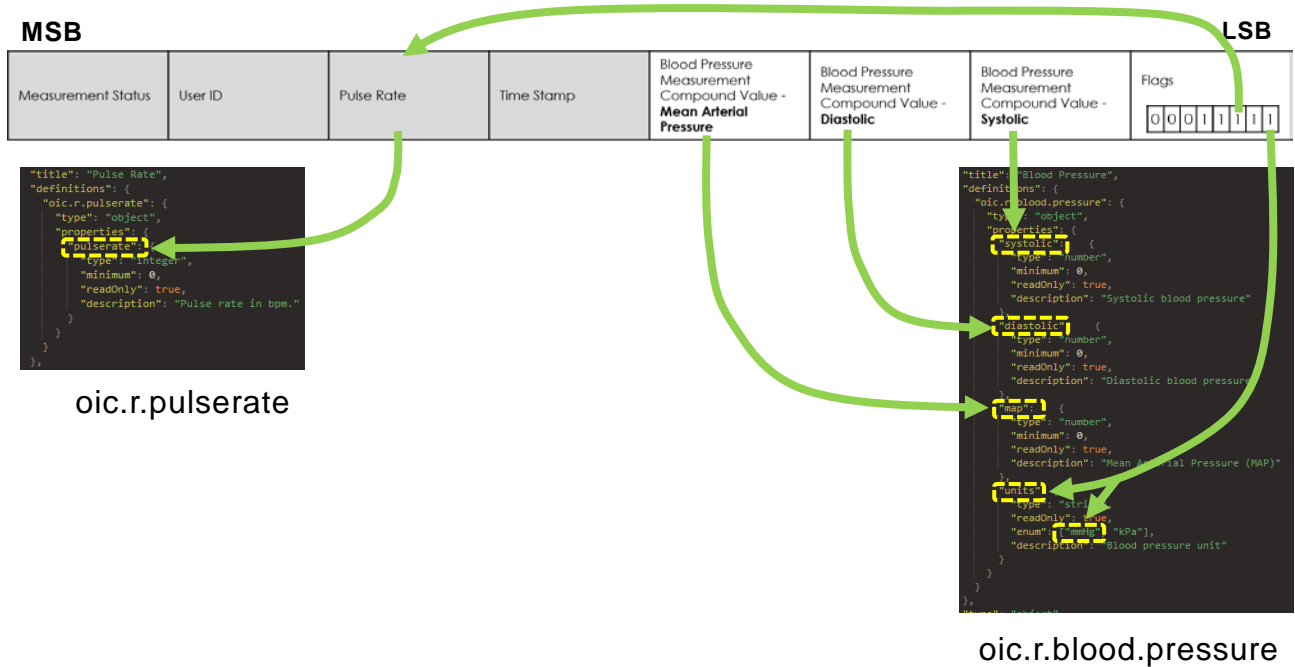
327

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 →	Blood Pressure Measurement ("org.bluetooth.characteristic.blood_pressure_measurement")	"oic.r.blood.pressure.systolic" "oic.r.blood.pressure.diastolic"

OCF Resource Property	"oic.r.blood.pressure.map"
	"oic.r.blood.pressure.units"
	"oic.r.pulserate.pulserate"

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



331
 332 **Figure 4 – An example for 1:N mapping between BLE Characteristic and OCF Properties**

333 **6.2.3.2 Translation for well-defined set**

334 **6.2.3.2.1 General**

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

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

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

			"oic.r.glucose.carb"	
			"oic.r.glucose.exercise"	
			"oic.r.glucose.hba1c"	
			"oic.r.glucose.health"	
			"oic.r.glucose.meal"	
			"oic.r.glucose.medication"	
			"oic.r.glucose.samplelocation"	
			"oic.r.glucose.tester"	
	Device Information Service		"oic.wk.d"	
			"oic.wk.p"	
Health Thermometer Profile	Health Thermometer Service	"oic.r.bodythermometer-am"	"oic.r.temperature"	"oic.d.bodythermometer"
			"oic.r.body.location.temperature"	
	Device Information Service		"oic.wk.d"	
			"oic.wk.p"	
Weight Scale Profile	Weight Scale Service	"oic.r.bodyscale-am"	"oic.r.weight"	"oic.d.bodyscale"
			"oic.r.bmi"	
			"oic.r.height"	
			"oic.r.body.fat"	
			"oic.r.body.water"	
			"oic.r.body.slm"	
	"oic.r.body.ffm"			
	Device Information Service		"oic.wk.d"	
		"oic.wk.p"		

338 **6.2.3.2.2 URI for Virtual OCF Resource**

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

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

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

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

- 346 – URI for Collection Resource: */<BLE Service name without prefix "org.bluetooth.service">* (e.g.
- 347 BLE Service "Health Thermometer (org.bluetooth.service.health_thermometer)":
- 348 /health_thermometer)
- 349 – URI for each OCF Resource link: */<OCF Resource Type value of corresponding linked*
- 350 *Resource without prefix "oic.r">* (e.g. /temperature for "oic.r.temperature",
- 351 /body.location.temperature for "oic.r.body.location.temperature")

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

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

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

Table 4 – URI mapping example

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

357

6.2.3.2.3 Common Properties of Resource Type

359 Resource Type ("rt", Mandatory): value of "rt" in corresponding OCF Resource specified in
360 ISO/IEC 30118-4:2019.

361 Interface ("if", Mandatory): value of "if" in corresponding OCF Resource specified in
362 ISO/IEC 30118-4:2019.

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

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

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

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

371 Spec Version ("icv", Mandatory): Spec version of ISO/IEC 30118-1:2019 that the Bridging Function
372 implements should be used.

373 Device ID ("di", Mandatory): as specified in ISO/IEC 30118-2:2019, the value of the "di" Property
374 of OCF Devices (including Virtual OCF Devices) shall be established as part of On-boarding of
375 that Virtual OCF Device.

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

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

6.2.3.3 Exposing a BLE GATT Server as a Virtual OCF Server

381 Table 5 shows how OCF Device Properties as specified in ISO/IEC 30118-1:2019, should be
382 derived, typically from fields specified in BLE Device Information Service (Spec Type:
383 "org.bluetooth.service.device_information", Service ID: 0x180A) and Generic Access Service
384 (Spec Type: "org.bluetooth.service.generic_access", Service ID: 0x1800).

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

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory ?	From BLE Device Service Characteristic value	BLE Description	BLE Mandatory ?
(Device) Name	"n"	Human friendly name For example, "Bob's Thermostat"	Y	Device Name (Generic Access)		Y
Spec Version	"icv"	Spec version of ISO/IEC 30118-1:2019 this Device is implemented to, The syntax is "core.major.minor"]	Y	(none)	Bridging Function should return its own value	
Device ID	"di"	Unique identifier for Device. This value shall be as defined in ISO/IEC 30118-2:2019 for DeviceID.	Y	(none)	Use as defined in ISO/IEC 30118-2:2019	
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	N	Manufacturer Name String (Device Information)	This characteristic represents the name of the	N

		object has a "language" field and a "value" field containing the manufacturer name in the indicated language.			manufacturer of the Device.	
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

386
387
388
389
390
391
392

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

Table 6 shows how platform Properties, as specified in ISO/IEC 30118-1:2019, are derived, typically from fields specified in BLE Device Information Service and Generic Access Service.

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

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory?	From BLE Device Service Characteristic value	BLE Description	BLE Mandatory?
Platform ID	"pi"	Unique identifier for the physical platform (UUID); 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

393
394
395
396
397

Table 7 shows how configurable OCF Platform Properties, as specified in Table 16 in ISO/IEC 30118-1:2019, 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.	

399

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

402 **6.2.3.4 On-the-fly Translation**

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

405 **6.2.3.5 Protocol translation between BLE and OCF**

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

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

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

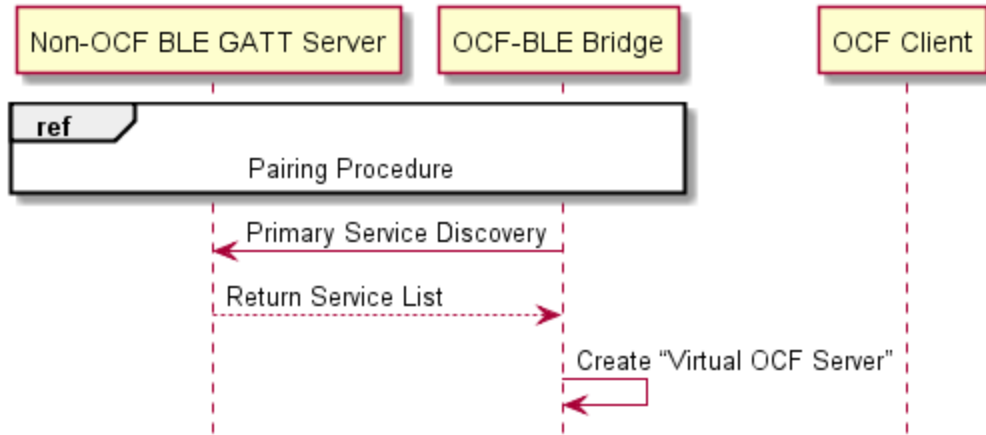
BLE GATT Feature	OCF CRUDN
N/A (Not Supported)	CREATE
Characteristic Value Read	RETRIEVE
Characteristic Value Write	UPDATE
N/A (Not Supported)	DELETE
Characteristic Descriptor Value Write	NOTIFY request
Characteristic Value Notification/Indication	NOTIFICATION response

418

419 **6.2.3.6 Illustrative OCF to BLE translation flows**

420 **6.2.3.6.1 Initialization**

421 Figure 5 shows the initial pairing procedure.



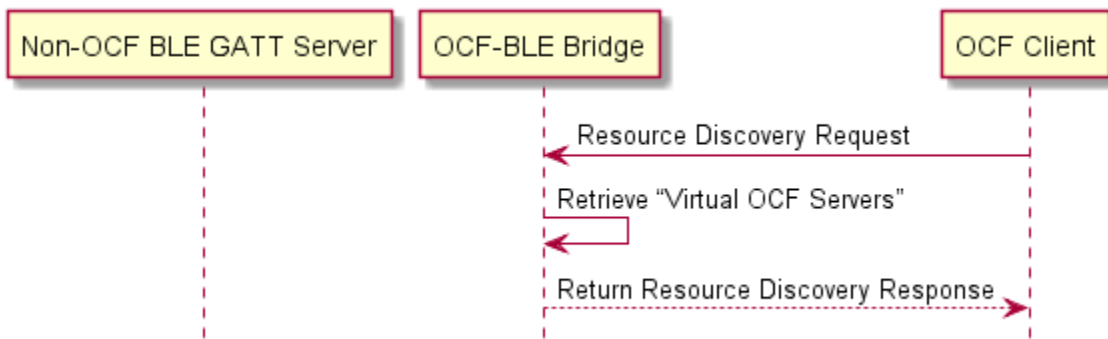
422

423

Figure 5 – Initialization

6.2.3.6.2 Resource Discovery

424 Figure 6 shows the resource discovery procedure.
425



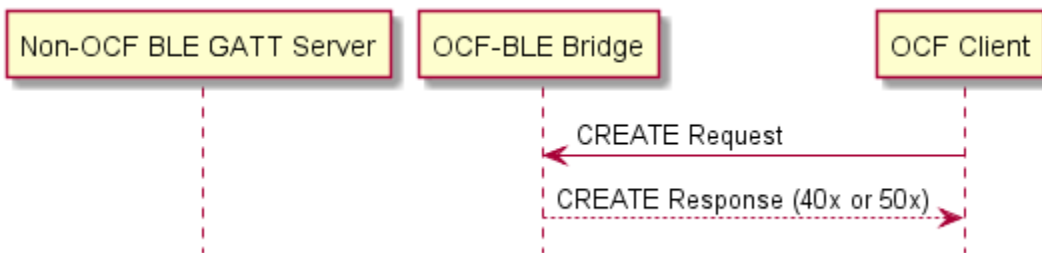
426

427

Figure 6 – Resource Discovery

6.2.3.6.3 Create Resource

428 Figure 7 illustrates Resource creation.
429



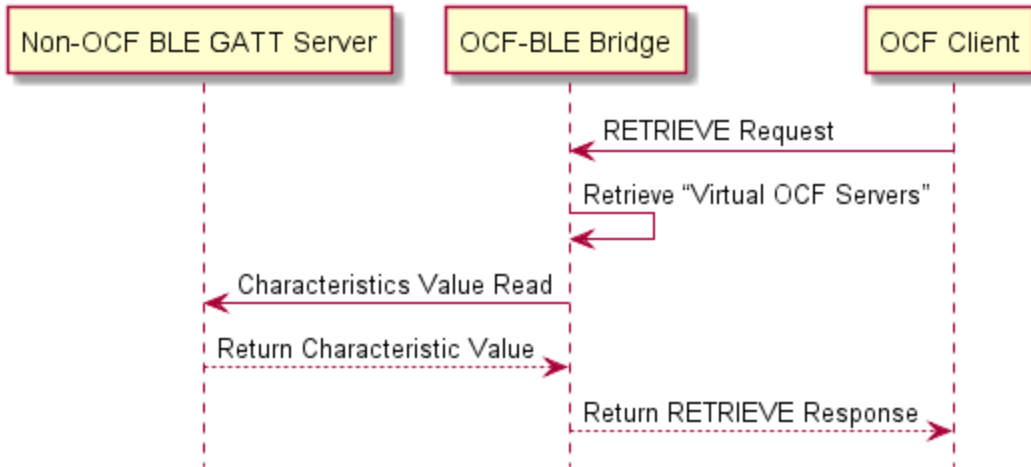
430

431

Figure 7 – Create Resource

6.2.3.6.4 Retrieve Resource

432 Figure 8 illustrates Resource RETRIEVAL.
433



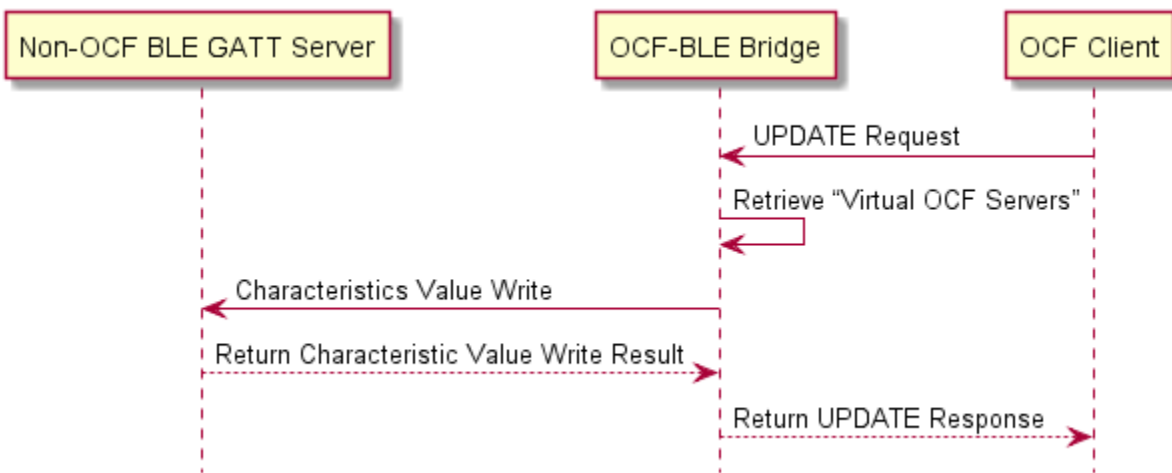
434

435

Figure 8 – Retrieve Resource

6.2.3.6.5 Update Resource

436 Figure 9 illustrates Resource UPDATE.
437



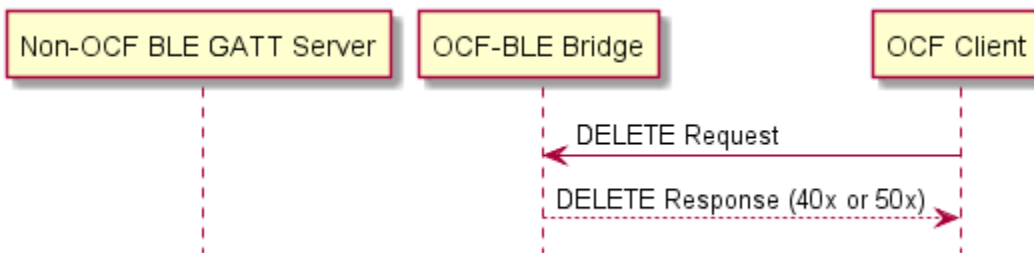
438

439

Figure 9 – Update Resource

6.2.3.6.6 Delete Resource

440 Figure 10 illustrates Resource DELETE. Note that this only applies to Resources that were created.
441



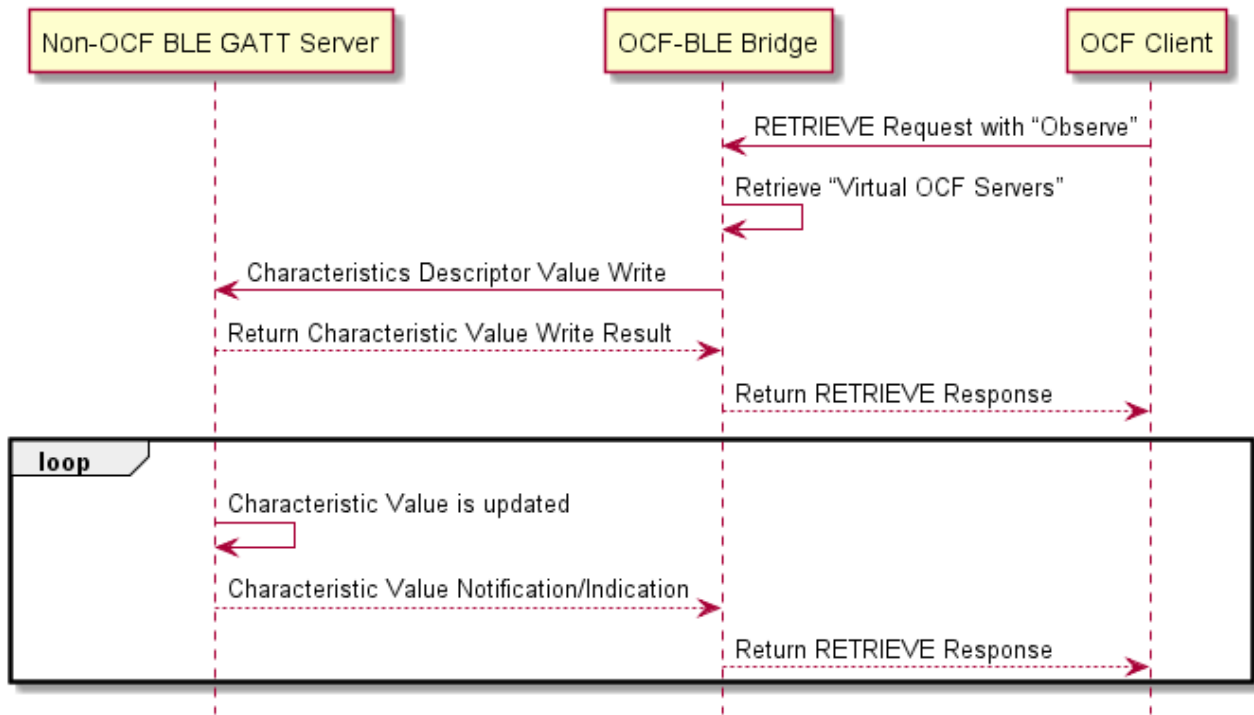
442

443

Figure 10 – Delete Resource

444 **6.2.3.6.7 Set Notification & Send Notification**

445 Figure 11 illustrates the establishment and sending of a notification.



446

447 **Figure 11 – Set Notification and send Notification**

448

449 **6.2.3.7 Error handling**

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

455 **7 Device Type Mapping**

456 **7.1 Introduction**

457 This clause contains the mappings to OCF Device Types.

458 **7.2 BLE Profile to OCF Device Types**

459 Table 9 captures the equivalency mapping between BLE Profile and OCF defined Device Types.
 460 The minimum Resource sets for each OCF Device is provided in ISO/IEC 30118-5:2019.

461 **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"

462

463 **8 BLE Profile to Resource Equivalence**

464 **8.1 Introduction**

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

467 **8.2 BLE Services to OCF Resources**

468 Table 10 captures the equivalency mapping between BLE Services and OCF defined Resource
 469 Types (see ISO/IEC 30118-4:2019). Detailed Property by Property mappings are provided in
 470 clause 9.

471

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"
	"oic.r.bodyscale-am"	"oic.r.weight"

Weight Scale Service		"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"

472

473 **9 Detailed Mappings**

474 **9.1 Introduction**

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

478 **9.2 Blood Pressure Mapping**

479 **9.2.1 Derived model**

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

481 **9.2.2 Property definition**

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

484 **Table 11 – The Property mapping for**
 485 **"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, #	N/A

		+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.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

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

488 **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

489 9.2.3 Derived model definition

```

490 {
491   "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.BLP.json#",
492   "$schema": "http://json-schema.org/draft-04/schema#",
493   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
494 reserved.",
495   "title": "Blood Pressure",
496   "definitions": {
497     "byte": {
498       "type": "integer",
499       "minimum": 0,
500       "maximum": 255
501     },
502     "byteArray": {
503       "type": "array",
504       "items": { "$ref": "#/definitions/byte" },
505       "minItems": 1,
506       "uniqueItems": false
507     },
508     "org.bluetooth.characteristic.blood_pressure_measurement" : {
509       "type": "object",
510       "properties": {
511         "blood_pressure_measurement[length - 3 : length - 1]": {
512           "$ref": "#/definitions/byteArray",
513           "description": "Blood Pressure Measurement Compound Value -
514 Systolic",
515           "x-ocf-conversion": {
516             "x-ocf-alias": "oic.r.blood.pressure",
517             "x-to-ocf": [
518               "def
519 ieee11073_Sfloat_2_Float(sfloat_value):",
520               "# reserved value for Infinity or NaN
521 (Not a Number)",
522               "reserved_float_values = {",
523                 "0x07FE:math.inf, # +INFINITY",
524                 "0x07FF:math.nan, # NaN (Not a
525 Number)",
526                 "0x0800:math.nan, # NRes (Not at this
527 Resolution)",
528                 "0x0801:math.nan, # Reserved for
529 future",
530                 "0x0802:-math.inf # -INFINITY",
531               "}",
532               "mantissa = sfloat_value & 0x0FFF",
533               "exponent = sfloat_value >> 12",
534               "if (exponent >= 0x0008):",
535                 "exponent = -((0x000F + 1) -
536 exponent)",
537               "output = 0",
538               "if (mantissa >= 0x07FE and mantissa <=
539 0x0802):",
540                 "output =
541 reserved_float_values[mantissa]",

```

```

543                                     "else:",
544                                     "if (mantissa >= 0x0800):",
545                                     "mantissa = -((0x0FFF + 1) -
546 mantissa)",
547                                     "magnitude = pow(10.0, exponent)",
548                                     "output = (mantissa * magnitude)",
549                                     "return output",
550 "length = len(blood_pressure_measurement)",
551 "flags = blood_pressure_measurement[length -
552 1]",
553                                     "oic.r.blood.pressure.systolic =
554 ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 3 : length - 1])",
555                                     "oic.r.blood.pressure.units = \"mmHg\" if
556 (flags & 0x01) else \"kPa\"",
557                                     ],
558 "x-from-ocf": [
559     "N/A"
560 ]
561     },
562 },
563 "blood_pressure_measurement[length - 5 : length - 3]": {
564     "$ref": "#/definitions/byteArray",
565     "description": "Blood Pressure Measurement Compound Value -
566 Diastolic",
567     "x-ocf-conversion": {
568         "x-ocf-alias": "oic.r.blood.pressure",
569         "x-to-ocf": [
570             "def
571 ieee11073_Sfloat_2_Float(sfloat_value):",
572                                     "# reserved value for Infinity or NaN
573 (Not a Number)",
574                                     "reserved_float_values = {",
575                                     "0x07FE:math.inf, # +INFINITY",
576                                     "0x07FF:math.nan, # NaN (Not a
577 Number)",
578                                     "0x0800:math.nan, # NRes (Not at this
579 Resolution)",
580                                     "0x0801:math.nan, # Reserved for
581 future",
582                                     "0x0802:-math.inf # -INFINITY",
583                                     }",
584 "mantissa = sfloat_value & 0x0FFF",
585 "exponent = sfloat_value >> 12",
586 "if (exponent >= 0x0008):",
587     "exponent = -((0x000F + 1) -
588 exponent)",
589     "output = 0",
590     "if (mantissa >= 0x07FE and mantissa <=
591 0x0802):",
592     "output =
593 reserved_float_values[mantissa]",
594     "else:",
595     "if (mantissa >= 0x0800):",
596     "mantissa = -((0x0FFF + 1) -
597 mantissa)",
598     "magnitude = pow(10.0, exponent)",
599     "output = (mantissa * magnitude)",
600     "return output",
601     "length = len(blood_pressure_measurement)",
602     "oic.r.blood.pressure.diastolic =
603 ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 5 : length - 3])"
604     ],
605 "x-from-ocf": [
606     "N/A"
607 ]
608     },
609 },
610 "blood_pressure_measurement[length - 7 : length - 5]": {
611     "$ref": "#/definitions/byteArray",
612     "description": "Blood Pressure Measurement Compound Value -
613 Mean Arterial Pressure",

```

```

614         "x-ocf-conversion": {
615             "x-ocf-alias": "oic.r.blood.pressure",
616             "x-to-ocf": [
617                 "def
618 ieee11073_Sfloat_2_Float(sfloat_value):",
619                                     "# reserved value for Infinity or NaN
620 (Not a Number)",
621                                     "reserved_float_values = {",
622                                     "0x07FE:math.inf, # +INFINITY",
623                                     "0x07FF:math.nan, # NaN (Not a
624 Number)",
625                                     "0x0800:math.nan, # NRes (Not at this
626 Resolution)",
627                                     "0x0801:math.nan, # Reserved for
628 future",
629                                     "0x0802:-math.inf # -INFINITY",
630                                     "}",
631                                     "mantissa = sfloat_value & 0x0FFF",
632                                     "exponent = sfloat_value >> 12",
633                                     "if (exponent >= 0x0008):",
634                                     "exponent = -((0x000F + 1) -
635 exponent)",
636                                     "output = 0",
637                                     "if (mantissa >= 0x07FE and mantissa <=
638 0x0802):",
639                                     "output =
640 reserved_float_values[mantissa]",
641                                     "else:",
642                                     "if (mantissa >= 0x0800):",
643                                     "mantissa = -((0x0FFF + 1) -
644 mantissa)",
645                                     "magnitude = pow(10.0, exponent)",
646                                     "output = (mantissa * magnitude)",
647                                     "return output",
648                                     "length = len(blood_pressure_measurement)",
649                                     "oic.r.blood.pressure.map =
650 ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 7 : length - 5])"
651                                     ],
652             "x-from-ocf": [
653                 "N/A"
654             ]
655         }
656     },
657     "blood_pressure_measurement[length - 7 - timestamp_len - 2 : length
658 - 7 - timestamp_len]": {
659         "$ref": "#/definitions/byteArray",
660         "description": "Pulse Rate",
661         "x-ocf-conversion": {
662             "x-ocf-alias": "oic.r.pulserate",
663             "x-to-ocf": [
664                 "def
665 ieee11073_Sfloat_2_Float(sfloat_value):",
666                                     "# reserved value for Infinity or NaN
667 (Not a Number)",
668                                     "reserved_float_values = {",
669                                     "0x07FE:math.inf, # +INFINITY",
670                                     "0x07FF:math.nan, # NaN (Not a
671 Number)",
672                                     "0x0800:math.nan, # NRes (Not at this
673 Resolution)",
674                                     "0x0801:math.nan, # Reserved for
675 future",
676                                     "0x0802:-math.inf # -INFINITY",
677                                     "}",
678                                     "mantissa = sfloat_value & 0x0FFF",
679                                     "exponent = sfloat_value >> 12",
680                                     "if (exponent >= 0x0008):",
681                                     "exponent = -((0x000F + 1) -
682 exponent)",
683                                     "output = 0",
684                                     "if (mantissa >= 0x07FE and mantissa <=

```

```

685 0x0802):",
686                                     "output =
687 reserved_float_values[mantissa]",
688                                     "else:",
689                                     "if (mantissa >= 0x0800):",
690                                     "    mantissa = -((0x0FFF + 1) -
691 mantissa)",
692                                     "    magnitude = pow(10.0, exponent)",
693                                     "    output = (mantissa * magnitude)",
694                                     "    return output",
695 "length = len(blood_pressure_measurement)",
696 "flags = blood_pressure_measurement[length -
697 1]",
698 "timestamp_len = 7 if (flags & 0x02) else 0",
699 "oic.r.pulserate.pulserate =
700 ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 7 - timestamp_len - 2 : length - 7 -
701 timestamp_len])"
702                                     ],
703                                     "x-from-ocf": [
704                                     "N/A"
705                                     ]
706                                     }
707                                     }
708                                     }
709                                     },
710                                     },
711                                     "type": "object",
712                                     "allOf": [
713                                     { "$ref": "#/definitions/byte" },
714                                     { "$ref": "#/definitions/byteArray" },
715                                     { "$ref": "#/definitions/org.bluetooth.characteristic.blood_pressure_measurement" }
716                                     ],
717                                     "required": [
718                                     "blood_pressure_measurement[length - 3 : length - 1]",
719                                     "blood_pressure_measurement[length - 5 : length - 3]"
720                                     ]
721                                     }
722                                     }
723                                     }
724                                     }
725                                     }

```

9.3 Glucose Measurement Mapping

9.3.1 Derived model

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

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

9.3.2 Property definition

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

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

BLE Property name	OCF Resource	To OCF	From OCF
glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len - 10]	oic.r.glucose	def ieee11073_Sfloat_2_Float(sfloat_value):# reserved value for Infinity or NaN (Not a Number)reserved_float_values = {0x07FE:math.inf, # +INFINITY0x07FF:math.nan, # NaN (Not a Number)0x0800:math.nan, # NRes (Not at this Resolution)0x0801:math.nan, # Reserved for future0x0802:-math.inf # -INFINITY}mantissa = sfloat_value & 0x0FFFexponent = sfloat_value >> 12if	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(glucose_measurement)flags = glucose_measurement[length - 1]timeoffset_len = 2 if (flags & 0x01) else 0if (flags & 0x02) == True: glucose = ieee11073_Sfloat_2_Float(glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len - 10]) oic.r.glucose.glucose = (glucose * 1000) if (flags & 0x04) else (glucose * 0.1 * 1000 * 1000) oic.r.glucose.units = "mmol/L" if (flags & 0x04) else "mg/dL"if (flags & 0x02) == False: oic.r.glucose.glucose = 0 oic.r.glucose.units = "mmol/L"	
glucose_measurement[length - 1 - 2 - timeoffset_len - 10]	oic.r.glucose.samplelocation	length = len(glucose_measurement)flags = glucose_measurement[length - 1]timeoffset_len = 2 if (flags & 0x01) else 0if (flags & 0x02): samplelocation = { 1:"finger", 2:"ast", 3:"earlobe", 4:"ctrlsolution" } oic.r.glucose.samplelocation.samplelocation = samplelocation[glucose_measurement[length - 1 - 2 - timeoffset_len - 10] & 0xf0]	N/A

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

736 **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

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

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

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

		(mantissa * magnitude) return outputlength = len(glucose_measurement_context) flags = glucose_measurement_context[length - 1] extflags_len = 1 if (flags & 0x80) else 0 carb_len = 3 if (flags & 0x01) else 0 if (flags & 0x01): oic.r.glucose.carb.carb = ieee11073_Sfloat_2_Float(glucose_measurement_context[length - carb_len - extflags_len - 3 : length - 1 - extflags_len - 3]) * 1000	
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 0 if (flags & 0x01): meal = { 1:"breakfast", 2:"lunch", 3:"dinner", 4:"snack", 5:"drink", 6:"supper", 7:"brunch" } oic.r.glucose.carb.meal = meal[glucose_measurement_context[length - 1 - extflags_len - 3]]	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 0 carb_len = 3 if (flags & 0x01) else 0 meal_len = 1 if (flags & 0x02) else 0 health_len = 1 if (flags & 0x04) else 0 if (flags & 0x08): oic.r.glucose.exercise.exercise = glucose_measurement_context[length - 2 - health_len - meal_len - carb_len - extflags_len - 3]	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, # +INFINITY 0x07FF:math.nan, # NaN (Not a Number) 0x0800:math.nan, # NRes (Not at this Resolution) 0x0801:math.nan, # Reserved for future 0x0802:-math.inf # -INFINITY} mantissa = sfloat_value & 0x0FFF exponent = sfloat_value >> 12 if (exponent >= 0x0008): exponent = -((0x000F + 1) - exponent) output = 0 if (mantissa >= 0x07FE and mantissa <= 0x0802): output = reserved_float_values[mantissa] else: if (mantissa >= 0x0800): mantissa = -((0x0FFF + 1) - mantissa) magnitude = pow(10.0, exponent) output = (mantissa * magnitude) return output length = len(glucose_measurement_context) flags = glucose_measurement_context[length - 1] extflags_len = 1 if (flags & 0x80) else 0 carb_len = 3 if (flags & 0x01) else 0 meal_len = 1 if (flags & 0x02) else 0 health_len = 1 if (flags & 0x04) else 0 exercise_len = 3 if (flags & 0x08) else 0 medication_len = 3 if (flags & 0x10) else 0 hba1c_len = 2 if (flags & 0x40) else 0 if (flags & 0x40): oic.r.glucose.hba1c.hba1c = ieee11073_Sfloat_2_Float(glucose_measurement_context[length - hba1c_len - medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3])	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 0 carb_len = 3 if (flags & 0x01) else 0 meal_len = 1 if (flags & 0x02) else 0 health_len = 1 if (flags & 0x04) else 0 if (flags & 0x04): health = { 1:"minor", 2:"major", 3:"menses", 4:"stress", 5:"none" } oic.r.glucose.health.health = health[glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3] & 0xf0] tester = { 1:"self", 2:"hcp", 3:"lab" } oic.r.glucose.tester.tester =	N/A

		tester[glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3] & 0x0f]	
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

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

743
744

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

745 **9.3.3 Derived model definition**

```

746 {
747   "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.GLP.json#",
748   "$schema": "http://json-schema.org/draft-04/schema#",
749   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
750 reserved.",
751   "title": "Glucose",
752   "definitions": {
753     "byte": {
754       "type": "integer",
755       "minimum": 0,
756       "maximum": 255
757     },
758     "byteArray": {
759       "type": "array",
760       "items": { "$ref": "#/definitions/byte" },
761       "minItems": 1,
762       "uniqueItems": false
763     },
764     "org.bluetooth.characteristic.glucose_measurement": {
765       "type": "object",
766       "properties": {
767         "glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len -
768 10]": {
769           "$ref": "#/definitions/byteArray",
770           "description": "Glucose Concentration",
771

```

```

772         "x-ocf-conversion": {
773             "x-ocf-alias": "oic.r.glucose",
774             "x-to-ocf": [
775                 "def ieee11073_Sfloat_2_Float(sfloat_value):",
776                 "# reserved value for Infinity or NaN (Not a Number)",
777                 "reserved_float_values = {",
778                 "    0x07FE:math.inf, # +INFINITY",
779                 "    0x07FF:math.nan, # NaN (Not a Number)",
780                 "    0x0800:math.nan, # NRes (Not at this Resolution)",
781                 "    0x0801:math.nan, # Reserved for future",
782                 "    0x0802:-math.inf # -INFINITY",
783                 "}",
784                 "mantissa = sfloat_value & 0x0FFF",
785                 "exponent = sfloat_value >> 12",
786                 "if (exponent >= 0x0008):",
787                 "    exponent = -((0x000F + 1) - exponent)",
788                 "output = 0",
789                 "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
790                 "    output = reserved_float_values[mantissa]",
791                 "else:",
792                 "    if (mantissa >= 0x0800):",
793                 "        mantissa = -((0x0FFF + 1) - mantissa)",
794                 "        magnitude = pow(10.0, exponent)",
795                 "        output = (mantissa * magnitude)",
796                 "return output",
797                 "length = len(glucose_measurement)",
798                 "flags = glucose_measurement[length - 1]",
799                 "timeoffset_len = 2 if (flags & 0x01) else 0",
800                 "if (flags & 0x02) == True:",
801                 "    glucose = ieee11073_Sfloat_2_Float(glucose_measurement[length - 2 -
802 timeoffset_len - 10 : length - timeoffset_len - 10])",
803                 "    oic.r.glucose.glucose = (glucose * 1000) if (flags & 0x04) else
804 (glucose * 0.1 * 1000 * 1000)",
805                 "    oic.r.glucose.units = 'mmol/L' if (flags & 0x04) else 'mg/dL'",
806                 "if (flags & 0x02) == False:",
807                 "    oic.r.glucose.glucose = 0",
808                 "    oic.r.glucose.units = 'mmol/L'"
809             ],
810             "x-from-ocf": [
811                 "N/A"
812             ]
813         }
814     },
815     "glucose_measurement[length - 1 - 2 - timeoffset_len - 10]": {
816         "$ref": "#/definitions/byteArray",
817         "description": "Sample Location",
818         "x-ocf-conversion": {
819             "x-ocf-alias": "oic.r.glucose.samplelocation",
820             "x-to-ocf": [
821                 "length = len(glucose_measurement)",
822                 "flags = glucose_measurement[length - 1]",
823                 "timeoffset_len = 2 if (flags & 0x01) else 0",
824                 "if (flags & 0x02):",
825                 "    samplelocation = { 1:'finger', 2:'ast', 3:'earlobe',
826 4:'ctrlsolution' }",
827                 "    oic.r.glucose.samplelocation.samplelocation =
828 samplelocation[glucose_measurement[length - 1 - 2 - timeoffset_len - 10] & 0xf0]"
829             ],
830             "x-from-ocf": [
831                 "N/A"
832             ]
833         }
834     }
835 },
836 ],
837
838 "org.bluetooth.characteristic.glucose_measurement_context": {
839     "type": "object",
840     "properties": {
841         "glucose_measurement_context[length - carb_len - extflags_len - 3 : length - 1 -
842 extflags_len - 3]": {

```

```

843     "$ref": "#/definitions/byteArray",
844     "description": "Carbohydrate",
845     "x-ocf-conversion": {
846         "x-ocf-alias": "oic.r.glucose.carb",
847         "x-to-ocf": [
848             "def ieee11073_Sfloat_2_Float(sfloat_value):",
849             "# reserved value for Infinity or NaN (Not a Number)",
850             "reserved_float_values = {",
851                 "0x07FE:math.inf, # +INFINITY",
852                 "0x07FF:math.nan, # NaN (Not a Number)",
853                 "0x0800:math.nan, # NRes (Not at this Resolution)",
854                 "0x0801:math.nan, # Reserved for future",
855                 "0x0802:-math.inf # -INFINITY",
856             "}",
857             "mantissa = sfloat_value & 0x0FFF",
858             "exponent = sfloat_value >> 12",
859             "if (exponent >= 0x0008):",
860                 "exponent = -((0x000F + 1) - exponent)",
861             "output = 0",
862             "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
863                 "output = reserved_float_values[mantissa]",
864             "else:",
865                 "if (mantissa >= 0x0800):",
866                     "mantissa = -((0x0FFF + 1) - mantissa)",
867                     "magnitude = pow(10.0, exponent)",
868                     "output = (mantissa * magnitude)",
869                 "return output",
870             "length = len(glucose_measurement_context)",
871             "flags = glucose_measurement_context[length - 1]",
872             "extflags_len = 1 if (flags & 0x80) else 0",
873             "carb_len = 3 if (flags & 0x01) else 0",
874             "if (flags & 0x01): ",
875             "    oic.r.glucose.carb.carb =
876 ieee11073_Sfloat_2_Float(glucose_measurement_context[length - carb_len - extflags_len - 3 : length
877 - 1 - extflags_len - 3]) * 1000"
878         ],
879         "x-from-ocf": [
880             "N/A"
881         ]
882     }
883 },
884 "glucose_measurement_context[length - 1 - extflags_len - 3]": {
885     "$ref": "#/definitions/byteArray",
886     "description": "Carbohydrate ID",
887     "x-ocf-conversion": {
888         "x-ocf-alias": "oic.r.glucose.carb",
889         "x-to-ocf": [
890             "length = len(glucose_measurement_context)",
891             "flags = glucose_measurement_context[length - 1]",
892             "extflags_len = 1 if (flags & 0x80) else 0",
893             "if (flags & 0x01): ",
894             "    meal = { 1:'breakfast', 2:'lunch', 3:'dinner', 4:'snack',
895 5:'drink', 6:'supper', 7:'brunch' }",
896             "    oic.r.glucose.carb.meal = meal[glucose_measurement_context[length -
897 1 - extflags_len - 3]]"
898         ],
899         "x-from-ocf": [
900             "N/A"
901         ]
902     }
903 },
904 "glucose_measurement_context[length - 2 - health_len - meal_len - carb_len -
905 extflags_len - 3]": {
906     "$ref": "#/definitions/byteArray",
907     "description": "Exercise Intensity",
908     "x-ocf-conversion": {
909         "x-ocf-alias": "oic.r.glucose.exercise",
910         "x-to-ocf": [
911             "length = len(glucose_measurement_context)",
912             "flags = glucose_measurement_context[length - 1]",
913             "extflags_len = 1 if (flags & 0x80) else 0",

```

```

914         "carb_len = 3 if (flags & 0x01) else 0",
915         "meal_len = 1 if (flags & 0x02) else 0",
916         "health_len = 1 if (flags & 0x04) else 0",
917         "if (flags & 0x08): ",
918         "    oic.r.glucose.exercise.exercise =
919 glucose_measurement_context[length - 2 - health_len - meal_len - carb_len - extflags_len - 3]"
920     ],
921     "x-from-ocf": [
922         "N/A"
923     ]
924     },
925 },
926 "glucose_measurement_context[length - hbalc_len - medication_len - exercise_len -
927 health_len - meal_len - carb_len - extflags_len - 3 : length - medication_len - exercise_len -
928 health_len - meal_len - carb_len - extflags_len - 3]": {
929     "$ref": "#/definitions/byteArray",
930     "description": "HbA1c",
931     "x-ocf-conversion": {
932         "x-ocf-alias": "oic.r.glucose.hbalc",
933         "x-to-ocf": [
934             "def ieee11073_Sfloat_2_Float(sfloat_value):",
935             "# reserved value for Infinity or NaN (Not a Number)",
936             "reserved_float_values = {",
937             "    0x07FE:math.inf, # +INFINITY",
938             "    0x07FF:math.nan, # NaN (Not a Number)",
939             "    0x0800:math.nan, # NRes (Not at this Resolution)",
940             "    0x0801:math.nan, # Reserved for future",
941             "    0x0802:-math.inf # -INFINITY",
942             "}",
943             "mantissa = sfloat_value & 0x0FFF",
944             "exponent = sfloat_value >> 12",
945             "if (exponent >= 0x0008):",
946             "    exponent = -((0x000F + 1) - exponent)",
947             "output = 0",
948             "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
949             "    output = reserved_float_values[mantissa]",
950             "else:",
951             "    if (mantissa >= 0x0800):",
952             "        mantissa = -((0x0FFF + 1) - mantissa)",
953             "        magnitude = pow(10.0, exponent)",
954             "        output = (mantissa * magnitude)",
955             "return output",
956             "length = len(glucose_measurement_context)",
957             "flags = glucose_measurement_context[length - 1]",
958             "extflags_len = 1 if (flags & 0x80) else 0",
959             "carb_len = 3 if (flags & 0x01) else 0",
960             "meal_len = 1 if (flags & 0x02) else 0",
961             "health_len = 1 if (flags & 0x04) else 0",
962             "exercise_len = 3 if (flags & 0x08) else 0",
963             "medication_len = 3 if (flags & 0x10) else 0",
964             "hbalc_len = 2 if (flags & 0x40) else 0",
965             "if (flags & 0x40):",
966             "    oic.r.glucose.hbalc.hbalc =
967 ieee11073_Sfloat_2_Float(glucose_measurement_context[length - hbalc_len - medication_len -
968 exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - medication_len -
969 exercise_len - health_len - meal_len - carb_len - extflags_len - 3])"
970         ],
971         "x-from-ocf": [
972             "N/A"
973         ]
974     }
975 },
976 "glucose_measurement_context[length - health_len - meal_len - carb_len -
977 extflags_len - 3]": {
978     "$ref": "#/definitions/byteArray",
979     "description": "Health, Tester",
980     "x-ocf-conversion": {
981         "x-ocf-alias": "oic.r.glucose.health",
982         "x-to-ocf": [
983             "length = len(glucose_measurement_context)",
984             "flags = glucose_measurement_context[length - 1]",

```

```

985         "extflags_len = 1 if (flags & 0x80) else 0",
986         "carb_len = 3 if (flags & 0x01) else 0",
987         "meal_len = 1 if (flags & 0x02) else 0",
988         "health_len = 1 if (flags & 0x04) else 0",
989         "if (flags & 0x04): ",
990         "    health = { 1:'minor', 2:'major', 3:'menses', 4:'stress',
991 5:'none' }",
992         "    oic.r.glucose.health.health =
993 health[glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3] &
994 0xf0]",
995         "    tester = { 1:'self', 2:'hcp', 3:'lab' }",
996         "    oic.r.glucose.tester.tester =
997 tester[glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3] &
998 0xf0]"
999     ],
1000     "x-from-ocf": [
1001         "N/A"
1002     ]
1003 }
1004 },
1005 "glucose_measurement_context[length - meal_len - carb_len - extflags_len - 3]": {
1006     "$ref": "#/definitions/byteArray",
1007     "description": "Meal",
1008     "x-ocf-conversion": {
1009         "x-ocf-alias": "oic.r.glucose.meal",
1010         "x-to-ocf": [
1011             "length = len(glucose_measurement_context)",
1012             "flags = glucose_measurement_context[length - 1]",
1013             "extflags_len = 1 if (flags & 0x80) else 0",
1014             "carb_len = 3 if (flags & 0x01) else 0",
1015             "meal_len = 1 if (flags & 0x02) else 0",
1016             "if (flags & 0x02): ",
1017             "    meal = { 1:'preprandial', 2:'postprandial', 3:'fasting',
1018 4:'casual', 5:'bedtime' }",
1019             "    oic.r.glucose.meal.meal = meal[glucose_measurement_context[length -
1020 meal_len - carb_len - extflags_len - 3]]"
1021         ],
1022         "x-from-ocf": [
1023             "N/A"
1024         ]
1025     }
1026 },
1027 "glucose_measurement_context[length - medication_len - exercise_len - health_len -
1028 meal_len - carb_len - extflags_len - 3 : length - 1 - exercise_len - health_len - meal_len -
1029 carb_len - extflags_len - 3]": {
1030     "$ref": "#/definitions/byteArray",
1031     "description": "Medication",
1032     "x-ocf-conversion": {
1033         "x-ocf-alias": "oic.r.glucose.mediasion",
1034         "x-to-ocf": [
1035             "def ieee11073_Sfloat_2_Float(sfloat_value):",
1036             "    # reserved value for Infinity or NaN (Not a Number)",
1037             "    reserved_float_values = {",
1038             "        0x07FE:math.inf, # +INFINITY",
1039             "        0x07FF:math.nan, # NaN (Not a Number)",
1040             "        0x0800:math.nan, # NRes (Not at this Resolution)",
1041             "        0x0801:math.nan, # Reserved for future",
1042             "        0x0802:-math.inf # -INFINITY",
1043             "    }",
1044             "    mantissa = sfloat_value & 0x0FFF",
1045             "    exponent = sfloat_value >> 12",
1046             "    if (exponent >= 0x0008):",
1047             "        exponent = -((0x000F + 1) - exponent)",
1048             "    output = 0",
1049             "    if (mantissa >= 0x07FE and mantissa <= 0x0802):",
1050             "        output = reserved_float_values[mantissa]",
1051             "    else:",
1052             "        if (mantissa >= 0x0800):",
1053             "            mantissa = -((0x0FFF + 1) - mantissa)",
1054             "            magnitude = pow(10.0, exponent)",
1055             "            output = (mantissa * magnitude)",

```

```

1056         "return output",
1057         "length = len(glucose_measurement_context)",
1058         "flags = glucose_measurement_context[length - 1]",
1059         "extflags_len = 1 if (flags & 0x80) else 0",
1060         "carb_len = 3 if (flags & 0x01) else 0",
1061         "meal_len = 1 if (flags & 0x02) else 0",
1062         "health_len = 1 if (flags & 0x04) else 0",
1063         "exercise_len = 3 if (flags & 0x08) else 0",
1064         "medication_len = 3 if (flags & 0x10) else 0",
1065         "hbalc_len = 2 if (flags & 0x40) else 0",
1066         "if (flags & 0x10): ",
1067         "    medication =
1068 ieeel1073_Sfloat_2_Float(glucose_measurement_context[length - medication_len - exercise_len -
1069 health_len - meal_len - carb_len - extflags_len - 3 : length - 1 - exercise_len - health_len -
1070 meal_len - carb_len - extflags_len - 3])",
1071         "    oic.r.glucose.medication.medication = medication * 1000",
1072         "    oic.r.glucose.medication.units = 'mL' if (flags & 0x20) else 'mg'"
1073     ],
1074     "x-from-ocf": [
1075         "N/A"
1076     ]
1077 }
1078 },
1079     "glucose_measurement_context[length - 1 - exercise_len - health_len - meal_len -
1080 carb_len - extflags_len - 3]": {
1081         "$ref": "#/definitions/byteArray",
1082         "description": "Medication ID",
1083         "x-ocf-conversion": {
1084             "x-ocf-alias": "oic.r.glucose.medication",
1085             "x-to-ocf": [
1086                 "length = len(glucose_measurement_context)",
1087                 "flags = glucose_measurement_context[length - 1]",
1088                 "extflags_len = 1 if (flags & 0x80) else 0",
1089                 "carb_len = 3 if (flags & 0x01) else 0",
1090                 "meal_len = 1 if (flags & 0x02) else 0",
1091                 "health_len = 1 if (flags & 0x04) else 0",
1092                 "exercise_len = 3 if (flags & 0x08) else 0",
1093                 "if (flags & 0x10): ",
1094                 "    regimen = { 1:'rapidacting', 2:'shortacting',
1095 3:'intermediateacting', 4:'longacting', 5:'premix' }",
1096                 "    oic.r.glucose.medication.regimen =
1097 regimen[ glucose_measurement_context[length - 1 - exercise_len - health_len - meal_len - carb_len -
1098 extflags_len - 3] ]"
1099             ],
1100             "x-from-ocf": [
1101                 "N/A"
1102             ]
1103         }
1104     }
1105 }
1106 },
1107 },
1108 "type": "object",
1109
1110 "allof": [
1111     { "$ref": "#/definitions/byte" },
1112     { "$ref": "#/definitions/byteArray" },
1113     { "$ref":
1114 "#/definitions/org.bluetooth.characteristic.org.bluetooth.characteristic.glucose_measurement" },
1115     { "$ref":
1116 "#/definitions/org.bluetooth.characteristic.org.bluetooth.characteristic.glucose_measurement_context" }
1117 ],
1118 "required": [
1119     "glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len -
1120 10]"
1121 ]
1122 }
1123 }
1124 }
1125 }
1126 }

```

1127 **9.4 Health Thermometer Mapping**

1128 **9.4.1 Derived model**

1129 The derived model: "org.bluetooth.characteristic.temperature_measurement".

1130 **9.4.2 Property definition**

1131 Table 17 provides the detailed per Property mapping for
1132 "org.bluetooth.characteristic.temperature_measurement".

1133 **Table 17 – The Property mapping for**
1134 **"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 = {0x007FFFE:math.inf, # +INFINITY0x007FFFF:math.nan, # NaN (Not a Number)0x0080000:math.nan, # NRes (Not at this Resolution)0x0080001:math.nan, # Reserved for future0x0080002:-math.inf # - INFINITY}mantissa = float_value & 0x0FFFFFFFexponent = float_value >> 24if (exponent >= 0x0000080):exponent = - ((0x00000FF + 1) - exponent)output = 0if (mantissa >= 0x007FFFE and mantissa <= 0x0080002):output = reserved_float_values[mantissa]else:if (mantissa >= 0x0080000):mantissa = - ((0x0FFFFFFF + 1) - mantissa)magnitude = pow(10.0, exponent)output = (mantissa * magnitude)return outputlength = len(temperature_measurement)flags = temperature_measurement[length - 1]oic.r.temperature.temperature = ieee11073_Float_2_Float(temperature_measurement[length - 5 : length - 1])oic.r.temperature.units = 'F' if (flags & 0x01) else 'C'	N/A
temperature_measurement[length - temperaturetype_len - timestamp_len - 5]	oic.r.body.location.temperature	length = len(temperature_measurement)flags = temperature_measurement[length - 1]timestamp_len = 7 if (flags & 0x02) 0temperaturetype_len = 1 if (flags & 0x04) 0if (flags & 0x04): bloc = { 1:'xxx', 2:'body', 3:'ear', 4:'finger', 5:'gastro', 6:'mouth', 7:'rectum', 8:'toe', 9:'tympanum' } oic.r.body.location.temperature.bloc = bloc[temperature_measurement[length - temperaturetype_len - timestamp_len - 5]]	N/A

1135 Table 18 provides the details of the Properties that are part of
1136 "org.bluetooth.characteristic.temperature_measurement".

1137 **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
---	--	----	------------------

1138 9.4.3 Derived model definition

```

1139 {
1140   "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.HTP.json#",
1141   "$schema": "http://json-schema.org/draft-04/schema#",
1142   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
1143 reserved.",
1144   "title": "Health Thermometer",
1145   "definitions": {
1146     "byte": {
1147       "type": "integer",
1148       "minimum": 0,
1149       "maximum": 255
1150     },
1151     "byteArray": {
1152       "type": "array",
1153       "items": { "$ref": "#/definitions/byte" },
1154       "minItems": 1,
1155       "uniqueItems": false
1156     },
1157     "org.bluetooth.characteristic.temperature_measurement" : {
1158       "type": "object",
1159       "properties": {
1160         "temperature_measurement[ length - 5 : length - 1 ]" : {
1161           "$ref": "#/definitions/byteArray",
1162           "description": "Temperature",
1163           "x-ocf-conversion": {
1164             "x-ocf-alias": "oic.r.temperature",
1165             "x-to-ocf": [
1166               "# convert IEEE11073 FLOAT to float",
1167               "def ieee11073_Float_2_Float(float_value):",
1168                 "# reserved value for Infinity or NaN
1169 (Not a Number)",
1170
1171               "reserved_float_values = {",
1172                 "0x007FFFFFFE:math.inf, # +INFINITY",
1173                 "0x007FFFFFFF:math.nan, # NaN (Not a
1174 Number)",
1175                 "0x00800000:math.nan, # NRes (Not at
1176 this Resolution)",
1177                 "0x00800001:math.nan, # Reserved for
1178 future",
1179                 "0x00800002:-math.inf # -INFINITY",
1180               "}",
1181               "mantissa = float_value & 0x00FFFFFF",
1182               "exponent = float_value >> 24",
1183               "if (exponent >= 0x00000080):",
1184                 "exponent = -((0x000000FF + 1) -
1185 exponent)",
1186               "output = 0",
1187               "if (mantissa >= 0x007FFFFFFE and mantissa
1188 <= 0x00800002):",
1189                 "output =
1190 reserved_float_values[mantissa]",
1191               "else:",
1192                 "if (mantissa >= 0x00800000):",
1193                   "mantissa = -((0x00FFFFFF + 1) -
1194 mantissa)",
1195                   "magnitude = pow(10.0, exponent)",
1196                   "output = (mantissa * magnitude)",
1197                 "return output",
1198               "length = len(temperature_measurement)",
1199               "flags = temperature_measurement[length -
1200 1]",
1201               "oic.r.temperature.temperature =
1202 ieee11073_Float_2_Float(temperature_measurement[ length - 5 : length - 1 ])",
1203               "oic.r.temperature.units = 'F' if (flags &

```



```

1204 0x01) else 'C'
1205
1206         ],
1207         "x-from-ocf": [
1208             "N/A"
1209         ]
1210     },
1211     "temperature_measurement[ length - temperaturetype_len -
1212 timestamp_len - 5 ]" : {
1213         "$ref": "#/definitions/byteArray",
1214         "description": "Temperature Type",
1215         "x-ocf-conversion": {
1216             "x-ocf-alias": "oic.r.body.location.temperature",
1217             "x-to-ocf": [
1218                 "length = len(temperature_measurement)",
1219                 "flags = temperature_measurement[length -
1220 1]",
1221                 "timestamp_len = 7 if (flags & 0x02) 0",
1222                 "temperaturetype_len = 1 if (flags & 0x04)
1223 0",
1224                 "if (flags & 0x04):",
1225                 "    bloc = { 1:'xxx', 2:'body', 3:'ear',
1226 4:'finger', 5:'gastro', 6:'mouth', 7:'rectum', 8:'toe', 9:'tympnum' }",
1227                 "    oic.r.body.location.temperature.bloc =
1228 bloc[temperature_measurement[ length - temperaturetype_len - timestamp_len - 5 ]]"
1229             ],
1230             "x-from-ocf": [
1231                 "N/A"
1232             ]
1233         }
1234     }
1235 }
1236 },
1237 },
1238 "type": "object",
1239
1240 "allOf": [
1241     { "$ref": "#/definitions/byte" },
1242     { "$ref": "#/definitions/byteArray" },
1243     { "$ref": "#/definitions/org.bluetooth.characteristic.temperature_measurement" }
1244 ],
1245
1246 "required": [
1247     "temperature_measurement[ length - 5 : length - 1 ]"
1248 ]
1249 }
1250 }
1251

```

1252 9.5 Weight Scale Mapping

1253 9.5.1 Derived model

1254 The derived model: "org.bluetooth.characteristic.weight_measurement".

1255 The derived model: "org.bluetooth.characteristic.body_composition_measurement".

1256 9.5.2 Property definition

1257 Table 19 provides the detailed per Property mapping for
1258 "org.bluetooth.characteristic.weight_measurement".

1259 **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 =	N/A

		int.from_bytes(weight_measurement[length - 3 : length - 1], 'big')oic.r.weight.units = 'lb' if (flags & 0x01) else 'kg'	
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

1260 Table 20 provides the details of the Properties that are part of
1261 "org.bluetooth.characteristic.weight_measurement".

1262 **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

1263 Table 21 provides the detailed per Property mapping for
1264 "org.bluetooth.characteristic.body_composition_measurement".

1265 **Table 21 – The Property mapping for**
1266 **"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]	oic.r.body.water	length = len(body_composition_measurement)flags_upperbyte = body_composition_measurement[length - 2]flags_lowerbyte = body_composition_measurement[length - 1]timestamp_len = 7 if (flags_lowerbyte & 0x02) else	N/A

- muscle_len - basal_len - userid_len - timestamp_len - 4]		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 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'	
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]	oic.r.body.slm	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'	N/A
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]	oic.r.body.ffm	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'	N/A

1267 Table 22 provides the details of the Properties that are part of
1268 "org.bluetooth.characteristic.body_composition_measurement".

1269 **Table 22 – The Properties of**
1270 **"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		no	Body Water Mass

- 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]			
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

1271 **9.5.3 Derived model definition**

```

1272 {
1273   "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.WSS.json#",
1274   "$schema": "http://json-schema.org/draft-04/schema#",
1275   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
1276 reserved.",
1277   "title": "Weight Scale",
1278   "definitions": {
1279     "byte": {
1280       "type": "integer",
1281       "minimum": 0,
1282       "maximum": 255
1283     },
1284     "byteArray": {
1285       "type": "array",
1286       "items": { "$ref": "#/definitions/byte" },
1287       "minItems": 1,
1288       "uniqueItems": false
1289     },
1290     "org.bluetooth.characteristic.weight_measurement" : {
1291       "type": "object",
1292       "properties": {
1293         "weight_measurement[length - 3 : length - 1]" : {
1294           "$ref": "#/definitions/byteArray",
1295           "description": "Weight",
1296           "x-ocf-conversion": {
1297             "x-ocf-alias": "oic.r.weight",
1298             "x-to-ocf": [
1299               "length = len(weight_measurement)",
1300               "flags = weight_measurement[length - 1]",
1301               "timeoffset_len = 7 if (flags & 0x02) else
1302 0",
1303               "oic.r.weight.weight =
1304 int.from_bytes(weight_measurement[length - 3 : length - 1], 'big')",
1305               "oic.r.weight.units = 'lb' if (flags & 0x01)
1306 else 'kg'"
1307             ],
1308             "x-from-ocf": [
1309               "N/A"
1310             ]
1311           }
1312         },
1313         "weight_measurement[length - 2 - userid_len - timeoffset_len - 3 :
1314 length - userid_len - timeoffset_len - 3]" : {
1315           "$ref": "#/definitions/byteArray",
1316           "description": "BMI",
1317           "x-ocf-conversion": {
1318             "x-ocf-alias": "oic.r.bmi",
1319             "x-to-ocf": [
1320

```

```

1321                                     "length = len(weight_measurement)",
1322                                     "flags = weight_measurement[length - 1]",
1323                                     "timeoffset_len = 7 if (flags & 0x02) else
1324 0",
1325                                     "userid_len = 1 if (flags & 0x04) else 0",
1326                                     "if (flags & 0x08):",
1327                                     "    oic.r.bmi.bmi =
1328 int.from_bytes(weight_measurement[length - 2 - userid_len - timeoffset_len - 3 : length -
1329 userid_len - timeoffset_len - 3], 'big')",
1330                                     ],
1331                                     "x-from-ocf": [
1332                                         "N/A"
1333                                     ]
1334                                 },
1335                                 "weight_measurement[length - height_len - userid_len -
1336 timeoffset_len - 3 : length - 2 - userid_len - timeoffset_len - 3]" : {
1337                                     "$ref": "#/definitions/byteArray",
1338                                     "description": "Height",
1339                                     "x-ocf-conversion": {
1340                                         "x-ocf-alias": "oic.r.height",
1341                                         "x-to-ocf": [
1342                                             "length = len(weight_measurement)",
1343                                             "flags = weight_measurement[length - 1]",
1344                                             "timeoffset_len = 7 if (flags & 0x02) else
1345 0",
1346                                             "userid_len = 1 if (flags & 0x04) else 0",
1347                                             "height_len = 4 if (flags & 0x08) else 0",
1348                                             "if (flags & 0x08):",
1349                                             "    oic.r.height.height =
1350 int.from_bytes(weight_measurement[length - height_len - userid_len - timeoffset_len - 3 : length -
1351 2 - userid_len - timeoffset_len - 3], 'big')",
1352                                             "    oic.r.height.units = 'in' if (flags &
1353 0x01) else 'm'",
1354                                         ],
1355                                         "x-from-ocf": [
1356                                             "N/A"
1357                                         ]
1358                                     }
1359                                 },
1360                                 }
1361                             },
1362                         },
1363                     "org.bluetooth.characteristic.body_composition_measurement" : {
1364                         "type": "object",
1365                         "properties": {
1366                             "body_composition_measurement[ length - 4 : length - 2]" : {
1367                                 "$ref": "#/definitions/byteArray",
1368                                 "description": "Body Fat Percentage",
1369                                 "x-ocf-conversion": {
1370                                     "x-ocf-alias": "oic.r.body.fat",
1371                                     "x-to-ocf": [
1372                                         "length = len(body_composition_measurement)",
1373                                         "oic.r.body.fat.bodyfat =
1374 int.from_bytes(body_composition_measurement[ length - 4 : length - 2], 'big')",
1375                                         "oic.r.body.fat.units = '%'",
1376                                     ],
1377                                     "x-from-ocf": [
1378                                         "N/A"
1379                                     ]
1380                                 }
1381                             },
1382                             "body_composition_measurement[ length - bwm_len - slm_len - ffm_len
1383 - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len - ffm_len -
1384 mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 ]" : {
1385                                 "$ref": "#/definitions/byteArray",
1386                                 "description": "Body Water Mass",
1387                                 "x-ocf-conversion": {
1388                                     "x-ocf-alias": "oic.r.body.water",
1389                                     "x-to-ocf": [
1390                                         "length = len(body_composition_measurement)",

```

```

1392                                     "flags_upperbyte =
1393 body_composition_measurement[length - 2]",
1394                                     "flags_lowerbyte =
1395 body_composition_measurement[length - 1]",
1396                                     "timestamp_len = 7 if (flags_lowerbyte &
1397 0x02) else 0",
1398                                     "userid_len = 1 if (flags_lowerbyte & 0x04)
1399 else 0",
1400                                     "basal_len = 2 if (flags_lowerbyte & 0x08)
1401 else 0",
1402                                     "muscle_len = 2 if (flags_lowerbyte & 0x10)
1403 else 0",
1404                                     "mm_len = 2 if (flags_lowerbyte & 0x20) else
1405 0",
1406                                     "ffm_len = 2 if (flags_lowerbyte & 0x40) else
1407 0",
1408                                     "slm_len = 2 if (flags_lowerbyte & 0x80) else
1409 0",
1410                                     "bwm_len = 2 if (flags_upperbyte & 0x01) else
1411 0",
1412                                     "if (flags_lowerbyte & 0x01): ",
1413                                     "    oic.r.body.water.bwater =
1414 int.from_bytes(body_composition_measurement[ length - bwm_len - slm_len - ffm_len - mm_len -
1415 muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len - ffm_len - mm_len -
1416 muscle_len - basal_len - userid_len - timestamp_len - 4 ], 'big')",
1417                                     "    oic.r.body.water.units = 'lb' if
1418 (flags_lowerbyte & 0x01) 'kg'"
1419                                     ],
1420                                     "x-from-ocf": [
1421                                         "N/A"
1422                                     ]
1423                                 },
1424                                 },
1425                                 "body_composition_measurement[ length - slm_len - ffm_len - mm_len -
1426 muscle_len - basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len - muscle_len -
1427 basal_len - userid_len - timestamp_len - 4 ]" : {
1428                                     "$ref": "#/definitions/byteArray",
1429                                     "description": "Soft Lean Mass",
1430                                     "x-ocf-conversion": {
1431                                         "x-ocf-alias": "oic.r.body.slm",
1432                                         "x-to-ocf": [
1433                                             "length = len(body_composition_measurement)",
1434                                             "flags_upperbyte =
1435 body_composition_measurement[length - 2]",
1436                                             "flags_lowerbyte =
1437 body_composition_measurement[length - 1]",
1438                                             "timestamp_len = 7 if (flags_lowerbyte &
1439 0x02) else 0",
1440                                             "userid_len = 1 if (flags_lowerbyte & 0x04)
1441 else 0",
1442                                             "basal_len = 2 if (flags_lowerbyte & 0x08)
1443 else 0",
1444                                             "muscle_len = 2 if (flags_lowerbyte & 0x10)
1445 else 0",
1446                                             "mm_len = 2 if (flags_lowerbyte & 0x20) else
1447 0",
1448                                             "ffm_len = 2 if (flags_lowerbyte & 0x40) else
1449 0",
1450                                             "slm_len = 2 if (flags_lowerbyte & 0x80) else
1451 0",
1452                                             "if (flags_lowerbyte & 0x01): ",
1453                                             "    oic.r.body.slm.bwater =
1454 int.from_bytes(body_composition_measurement[ length - slm_len - ffm_len - mm_len - muscle_len -
1455 basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len - muscle_len - basal_len -
1456 userid_len - timestamp_len - 4 ], 'big')",
1457                                             "    oic.r.body.slm.units = 'lb' if
1458 (flags_lowerbyte & 0x01) 'kg'"
1459                                             ],
1460                                             "x-from-ocf": [
1461                                                 "N/A"
1462                                             ]

```

```

1463     }
1464     },
1465     "body_composition_measurement[ length - ffm_len - mm_len -
1466 muscle_len - basal_len - userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len
1467 - userid_len - timestamp_len - 4 ]" : {
1468     "$ref": "#/definitions/byteArray",
1469     "description": "Fat Free Mass",
1470     "x-ocf-conversion": {
1471         "x-ocf-alias": "oic.r.body.ffm",
1472         "x-to-ocf": [
1473             "length = len(body_composition_measurement)",
1474             "flags_upperbyte =
1475 body_composition_measurement[length - 2]",
1476             "flags_lowerbyte =
1477 body_composition_measurement[length - 1]",
1478             "timestamp_len = 7 if (flags_lowerbyte &
1479 0x02) else 0",
1480             "userid_len = 1 if (flags_lowerbyte & 0x04)
1481 else 0",
1482             "basal_len = 2 if (flags_lowerbyte & 0x08)
1483 else 0",
1484             "muscle_len = 2 if (flags_lowerbyte & 0x10)
1485 else 0",
1486             "mm_len = 2 if (flags_lowerbyte & 0x20) else
1487 0",
1488             "ffm_len = 2 if (flags_lowerbyte & 0x40) else
1489 0",
1490             "if (flags_lowerbyte & 0x01): ",
1491             "    oic.r.body.ffm.bwater =
1492 int.from_bytes(body_composition_measurement[ length - ffm_len - mm_len - muscle_len - basal_len -
1493 userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len - userid_len -
1494 timestamp_len - 4 ], 'big')",
1495             "    oic.r.body.ffm.units = 'lb' if
1496 (flags_lowerbyte & 0x01) 'kg' "
1497         ],
1498         "x-from-ocf": [
1499             "N/A"
1500         ]
1501     }
1502     }
1503     }
1504     },
1505     },
1506     "type": "object",
1507     "allOf": [
1508         { "$ref": "#/definitions/byte" },
1509         { "$ref": "#/definitions/byteArray" },
1510         { "$ref": "#/definitions/org.bluetooth.characteristic.weight_measurement" },
1511         { "$ref": "#/definitions/org.bluetooth.characteristic.body_composition_measurement" }
1512     ],
1513     "required": [
1514         "weight_measurement[length - 3 : length - 1]"
1515     ]
1516 }
1517 }
1518 }
1519 }
1520

```

1521
1522
1523

Annex A (Informative) BLE GATT based Data Model

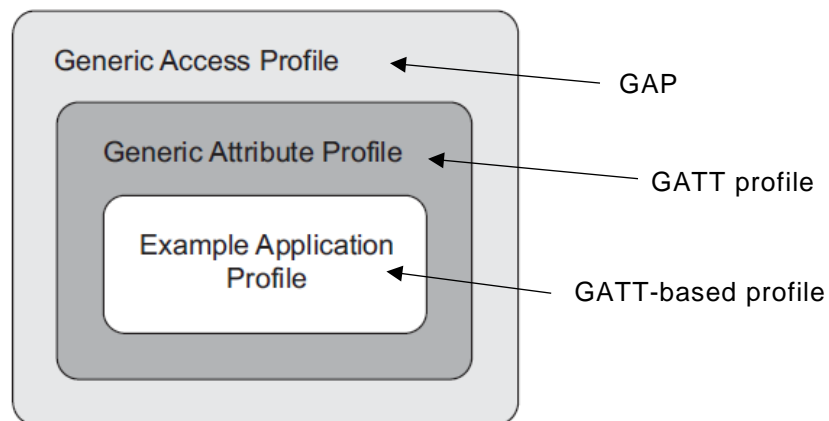
1524 **A.1 BLE GATT based data model & GATT features**

1525 **A.1.1 Introduction**

1526 The Generic Attribute Profile (GATT) defines a service framework using the Attribute Protocol.
1527 This framework defines procedures and formats of services and their characteristics. The
1528 procedures defined include discovering, reading, writing, notifying and indicating characteristics,
1529 as well as configuring the broadcast of characteristics.

1530 **A.1.2 Profile dependency**

1531 Figure A-1 depicts the structure and the dependencies of the profiles. A profile is dependent upon
1532 another profile if it re-uses parts of that profile by implicitly or explicitly referencing it.



1533
1534

Figure A-1 – profile dependencies

1535 **A.1.3 Configurations and roles**

1536 There are two roles defined in GATT profile:

- 1537 • Client: This is the device that initiates commands and requests towards the server and can
1538 receive responses, indications and notifications sent by the server.

- 1539 • Server: This is the device that accepts incoming commands and requests from the client
1540 and sends responses, indications and notifications to a client.

1541 A device can act in both roles at the same time.

1542 **A.1.4 GATT profile hierarchy**

1543 **A.1.4.1 Introduction**

1544 The GATT Profile specifies the structure in which profile data is exchanged. This structure defines
1545 basic elements such as services and characteristics, used in a profile. All of the elements are
1546 contained by Attributes. Attributes used in the ATT are containers that carry this profile data.

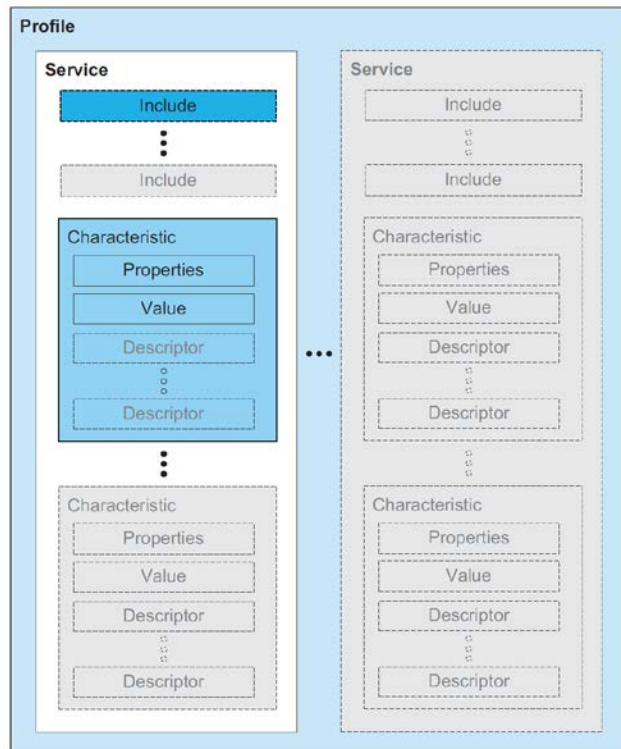
1547 The top level of the hierarchy is a profile. A profile is composed of one or more services necessary
1548 to fulfil a use case. A service is composed of characteristics or references to other services. Each
1549 characteristic contains a value and may contain optional information about the value. The service

1550 and characteristic and the components of the characteristic (i.e. value and descriptors) contain the
 1551 profile data and are all stored in Attributes on the server.

1552 Under GATT profile, entity that provides Service-Characteristics data model plays “server” role
 1553 and entity that gets data from GATT server plays “client” role.

1554 There are other application profiles based on GATT profile. They are called “GATT-based profiles”.

1555 Figure A-2 Illustrates the GATT profile hierarchy.



1556

1557

Figure A-2 – GATT profile hierarchy

1558 **A.1.4.2 Characteristic**

1559 A characteristic is a value used in a service along with properties and configuration information
 1560 about how the value is accessed and information about how the value is displayed or represented.
 1561 In GATT, a characteristic is defined by its characteristic definition. A characteristic definition
 1562 contains a characteristic declaration, characteristic properties, and a value and may contain
 1563 descriptors that describe the value or permit configuration of the server with respect to the
 1564 characteristic.

1565 **A.1.4.3 GATT features**

1566 GATT profile also supports GATT features. GATT feature defines how GATT-based data
 1567 exchanges take place. Each feature is mapped to one or more sub-procedures. These sub-
 1568 procedures describe how the ATT is used to accomplish the corresponding feature, please see
 1569 Table A-1.

1570

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

1571

1572
1573
1574

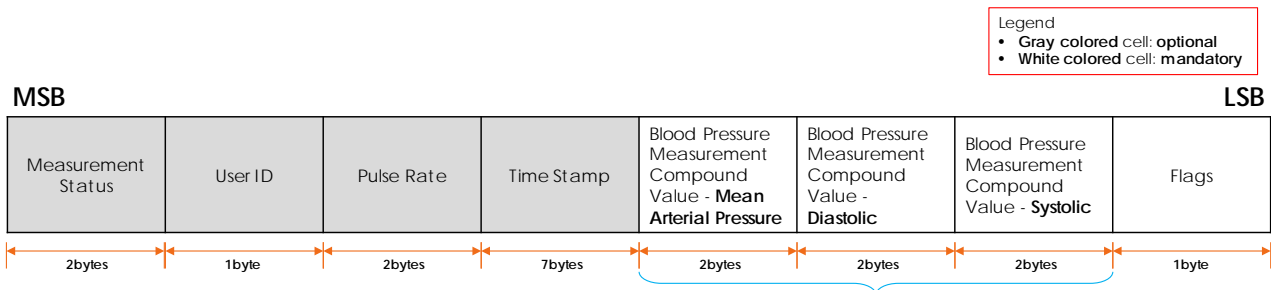
Annex B (Informative) Supporting Atomic Measurement Operation in BLE

1575 B.1 Atomic Measurement Resource Type in OCF

1576 Most OCF healthcare devices adopt the Atomic Measurement feature. Atomic Measurement
1577 Resource Type is a specialisation of a Collection to ensure that the Client can only access the
1578 Properties of the linked Resources as a single group. Thus, if an OCF device corresponding to a
1579 BLE device implements Atomic Measurement Resource Type, the BLE Bridging Function should
1580 guarantee that BLE GATT Characteristic values corresponding to properties of the Atomic
1581 Measurement Resource Type can be retrieved in atomic way.

1582 B.2 Case 1. One Characteristic covers all properties of an Atomic Measurement 1583 Resource Type

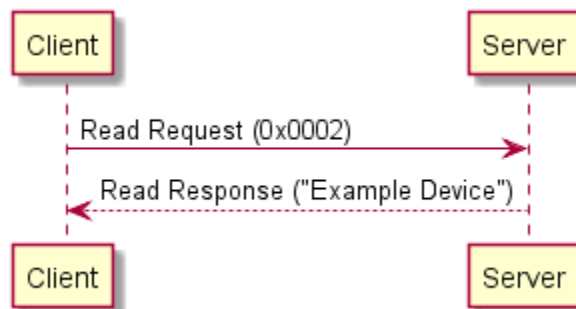
1584 In OCF-BLE mapping, a Service can be mapped to multiple OCF Resources and “a Characteristic”
1585 in a Service can be mapped to Properties in multiple OCF Resources. In general, “Value of a
1586 Characteristic” is a byte stream (see Figure B-1, byte stream is a value of “blood pressure
1587 measurement Characteristic”). Usually “value of a Characteristic” includes multiple fields like below
1588 example and each field can be mapped to a property of OCF Resource.



1589

1590 **Figure B-1 – Value of blood pressure measurement Characteristic**

1591 For blood pressure device, “blood pressure measurement Characteristic” can cover all properties
1592 in bloodpressuremonitor-am. So if BLE GATT client (OCF-BLE Bridge Platform) uses “Read
1593 Characteristic Value” operation, it can get all values corresponding to all properties in
1594 bloodpressuremonitor-am at one time (atomic operation); see Figure B-2 for an example flow.



1595

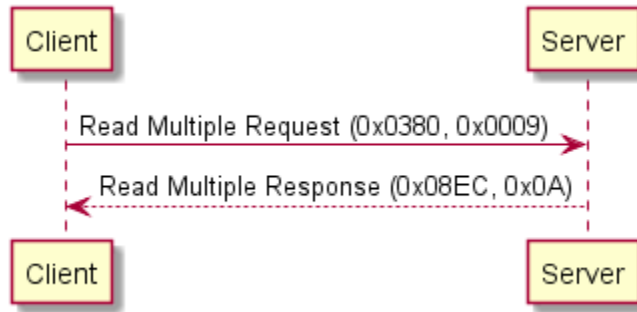
1596

1597

Figure B-2 – Read characteristic value example

1598 **B.3 Case 2. Multiple Characteristics cover all properties of an Atomic Measurement**
 1599 **Resource Type**

1600 For glucose meter, 2 Characteristics (glucose measurement Characteristic, glucose measurement
 1601 context Characteristic) cover all properties in glucosemeter-am. In this case, a BLE GATT client
 1602 (OCF-BLE Bridge Platform) can use “Read Multiple Characteristic Values” operation to get multiple
 1603 Characteristic values at one time; please see Figure B-3 for an example flow.



1604
 1605 **Figure B-3 – Read multiple characteristics value example**

1606 However, some BLE GATT server may not support all operations except for “Notification”. In this
 1607 case, a Characteristic value includes “sequence number” field, so BLE GATT client (OCF-BLE
 1608 Bridge Platform) can make a set of values which are measured at the same time by using it.

1609 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

1610
 1611 **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

1612
 1613 **Figure B-5 – Value of glucose measurement context Characteristic**