

OCF Resource to BLE Mapping Specification

VERSION 2.2.1 | December 2020



OPEN CONNECTIVITY
FOUNDATION™

CONTACT admin@openconnectivity.org

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

Legal Disclaimer

3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

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

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

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

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

21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63

CONTENTS

- Introduction..... vi
- 1 Scope..... 1
- 2 Normative references 1
- 3 Terms, definitions, symbols and abbreviated terms..... 1
 - 3.1 Terms and definitions..... 1
 - 3.2 Symbols and abbreviated terms 2
- 4 Document conventions and organization..... 2
 - 4.1 Conventions..... 2
 - 4.2 Notation 2
- 5 Theory of Operation..... 3
 - 5.1 Interworking Approach 3
 - 5.2 Mapping Syntax 3
- 6 BLE Translation..... 4
 - 6.1 Operational Scenarios 4
 - 6.1.1 Introduction 4
 - 6.1.2 Use case for BLE Bridging..... 4
 - 6.2 Requirements specific to BLE Bridging Function 5
 - 6.2.1 General 5
 - 6.2.2 Requirements specific to BLE 5
 - 6.2.3 Exposing BLE GATT Servers to OCF Clients 5
- 7 Device Type Mapping 16
 - 7.1 Introduction..... 16
 - 7.2 BLE Profile to OCF Device Types 16
- 8 BLE Profile to Resource Equivalence 17
 - 8.1 Introduction..... 17
 - 8.2 BLE Services to OCF Resources 17
- 9 Detailed Mappings..... 18
 - 9.1 Introduction..... 18
 - 9.2 Blood Pressure Mapping..... 18
 - 9.2.1 Derived model 18
 - 9.2.2 Property definition 18
 - 9.2.3 Derived model definition 20
 - 9.3 Glucose Measurement Mapping..... 23
 - 9.3.1 Derived model 23
 - 9.3.2 Property definition 23
 - 9.3.3 Derived model definition 27
 - 9.4 Health Thermometer Mapping..... 32
 - 9.4.1 Derived model 32
 - 9.4.2 Property definition 33
 - 9.4.3 Derived model definition 34
 - 9.5 Weight Scale Mapping 35
 - 9.5.1 Derived model 35

64	9.5.2	Property definition	35
65	9.5.3	Derived model definition	38
66	Annex A (Informative) BLE GATT based Data Model		42
67	A.1	BLE GATT based data model & GATT features.....	42
68	A.1.1	Introduction	42
69	A.1.2	Profile dependency.....	42
70	A.1.3	Configurations and roles.....	42
71	A.1.4	GATT profile hierarchy.....	42
72	Annex B (Informative) Supporting Atomic Measurement Operation in BLE		46
73	B.1	Atomic Measurement Resource Type in OCF	46
74	B.2	Case 1. One Characteristic covers all properties of an Atomic Measurement	
75		Resource Type	46
76	B.3	Case 2. Multiple Characteristics cover all properties of an Atomic Measurement	
77		Resource Type	47
78			

Figures

80	Figure 1 – OCF-BLE Bridge Platform Components.....	4
81	Figure 2 – BLE Bridging use case in real life.....	5
82	Figure 3 – Translation mapping rule illustration.....	6
83	Figure 4 – An example for 1:N mapping between BLE Characteristic and OCF Properties	7
84	Figure 5 – Initialization.....	13
85	Figure 6 – Resource Discovery	14
86	Figure 7 – Create Resource	14
87	Figure 8 – Retrieve Resource	14
88	Figure 9 – Update Resource	15
89	Figure 10 – Delete Resource	15
90	Figure 11 – Set Notification and send Notification.....	16
91	Figure A-1 – profile dependencies	42
92	Figure A-2 – GATT profile hierarchy.....	43
93	Figure B-1 – Value of blood pressure measurement Characteristic	46
94	Figure B-2 – Read characteristic value example	46
95	Figure B-3 – Read multiple characteristics value example.....	47
96	Figure B-4 – Value of glucose measurement Characteristic	47
97	Figure B-5 – Value of glucose measurement context Characteristic	47
98		

Tables

100 Table 1 – Translation rule between BLE and OCF data model.....5

101 Table 2 – BLE to OCF translation example (Blood Pressure Device).....6

102 Table 3 – BLE GATT-based Profile – OCF Resource mapping7

103 Table 4 – URI mapping example8

104 Table 5 – "oic.wk.d" Resource Type definition.....9

105 Table 6 – "oic.wk.p" Resource Type definition..... 11

106 Table 7 – "oic.wk.con.p" Resource Type definition 12

107 Table 8 – Protocol translation rule between BLE and OCF 13

108 Table 9 – BLE Profile to OCF Device Type Mapping 16

109 Table 10 – BLE Services to OCF Resource Type Mapping..... 17

110 Table 11 – The Property mapping for
111 "org.bluetooth.characteristic.blood_pressure_measurement". 18

112 Table 12 – The Properties of "org.bluetooth.characteristic.blood_pressure_measurement". ... 19

113 Table 13 – The Property mapping for "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". 26

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

131

Introduction

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

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

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

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

150 **1 Scope**

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

153 **2 Normative references**

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

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

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

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

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

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

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

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

183 Derived Models for Interoperability between IoT Ecosystems, Stevens & Merriam, March 2016
184 [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)
185 [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)

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

188 **3 Terms, definitions, symbols and abbreviated terms**

189 **3.1 Terms and definitions**

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

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

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

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

196 **3.1.1**

197 **GATT-based Profile**

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

199 **3.2 Symbols and abbreviated terms**

200 ATT Attribute protocol

201 GAP Generic Access Profile

202 GATT Generic Attribute profile

203 **4 Document conventions and organization**

204 **4.1 Conventions**

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

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

212 **4.2 Notation**

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

215 Required (or shall or mandatory).

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

219 Recommended (or should).

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

227 Allowed (or allowed).

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

231 Conditionally allowed (CA)

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

234 Conditionally required (CR)

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

238 DEPRECATED

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

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

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

247 5 Theory of Operation

248 5.1 Interworking Approach

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

252 5.2 Mapping Syntax

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

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

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

```
260 "<BLE Service Name>" : {  
261     "type": "object",  
262     "properties": {  
263         "<a value field in BLE Characteristic value>" : {  
264             "x-ocf-conversion" : {  
265                 "x-ocf-alias": "<corresponding OCF Resource type>",  
266                 "x-to-ocf": [  
267                     ...  
268                     ...  
269                 ],  
270                 "x-from-ocf": [  
271                     "N/A"  
272                 ]  
273             }  
274         }  
275     }  
276 }
```

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

- 279 – <a value field in BLE Characteristic value>: a Characteristic value is byte stream which is
280 composed of multiple value fields. “A value field in BLE Characteristic value” is a description
281 for one of them.
- 282 – <corresponding OCF Resource type>: an OCF Resource type which is corresponding to this
283 BLE Service.
- 284 – “N/A”: in BLE Bridging, most of the BLE devices are read only. So there is no specific value to
285 be written to the BLE devices from OCF Devices. Therefore, nothing is described in “x-from-
286 ocf” translation clause. “N/A” is used to describe this case.

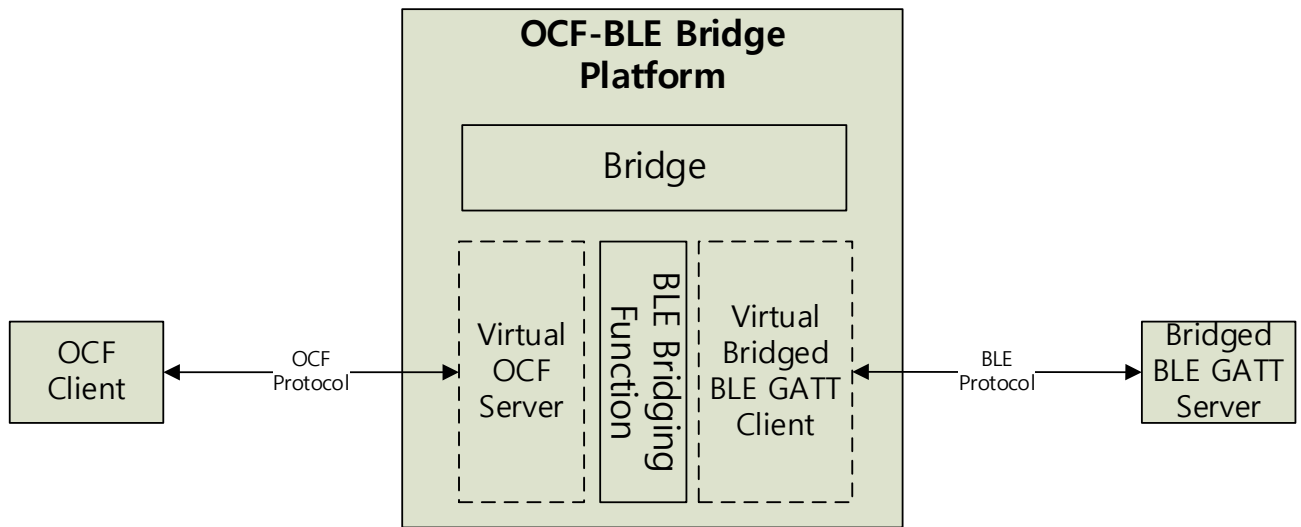
287 **6 BLE Translation**

288 **6.1 Operational Scenarios**

289 **6.1.1 Introduction**

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

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

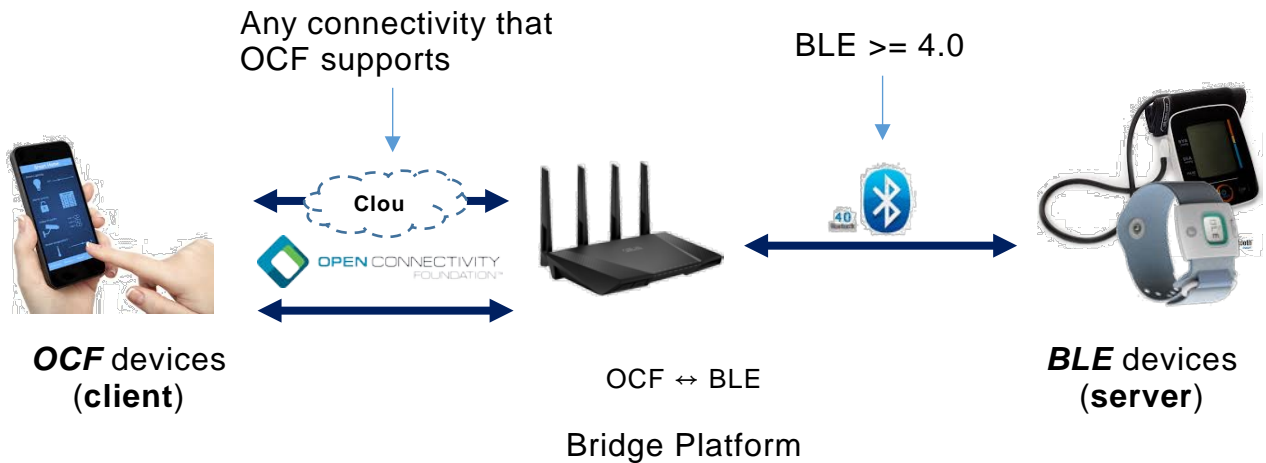


296

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

298 **6.1.2 Use case for BLE Bridging**

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



304

305

Figure 2 – BLE Bridging use case in real life

6.2 Requirements specific to BLE Bridging Function

6.2.1 General

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

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

6.2.2 Requirements specific to BLE

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

6.2.3 Exposing BLE GATT Servers to OCF Clients

6.2.3.1 General

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

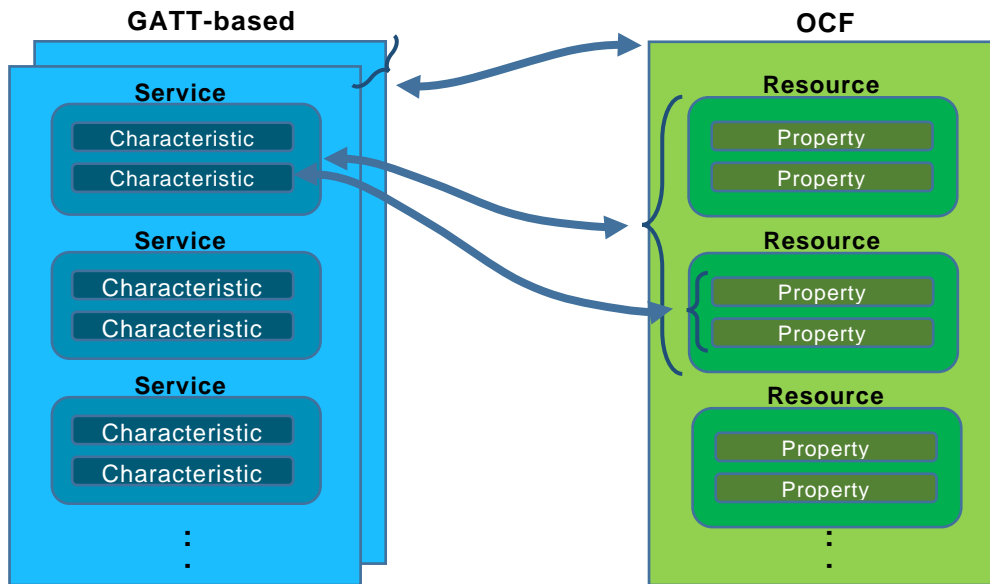
Basic translation rule between BLE Service/Characteristic model and OCF Resource model is described in Table 1.

Table 1 – Translation rule between BLE and OCF data model

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

323

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



332
 333

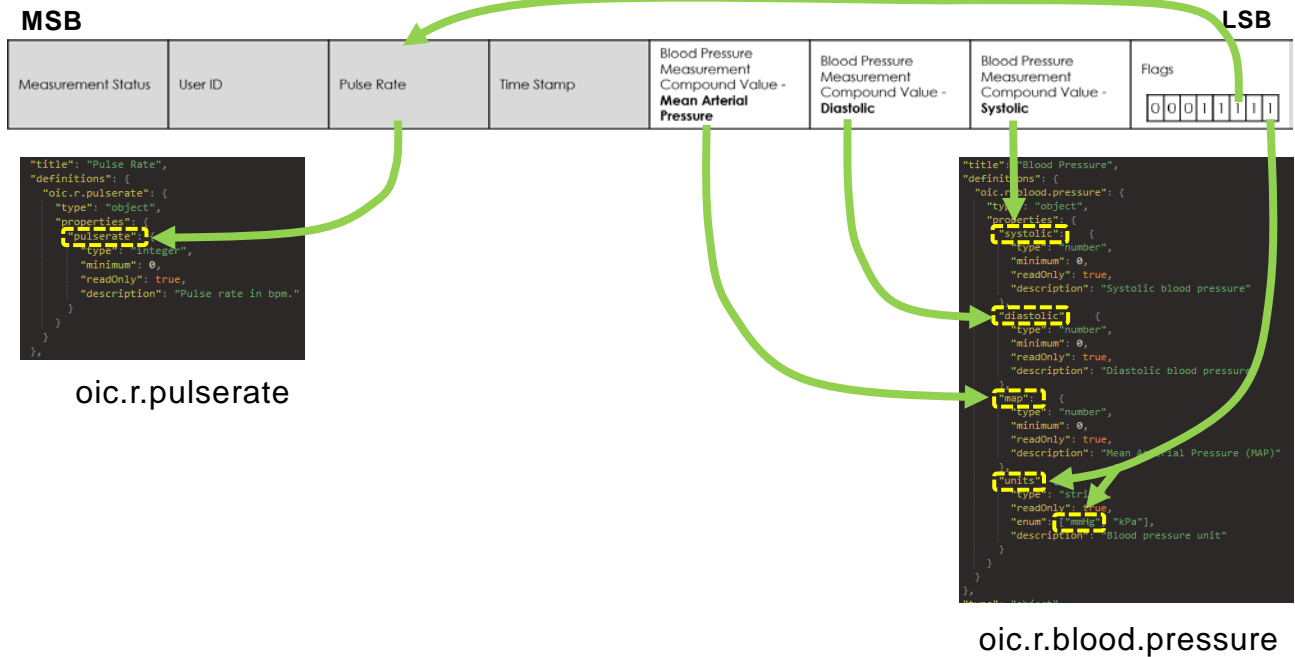
Figure 3 – Translation mapping rule illustration

334

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

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

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



338

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

340 **6.2.3.2 Translation for well-defined set**

341 **6.2.3.2.1 General**

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

344 **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"	
	Device Information Service		"oic.wk.d"	
			"oic.wk.p"	

345 **6.2.3.2.2 URI for Virtual OCF Resource**

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

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

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

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

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

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

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

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

363

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")

364

365 **6.2.3.2.3 Common Properties of Resource Type**

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

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

370 **6.2.3.2.4 Platform Resource ("rt" of "oic.wk.p")**

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

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

377 **6.2.3.2.5 Device Resource ("rt" of "oic.wk.d")**

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

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

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

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

387 **6.2.3.3 Exposing a BLE GATT Server as a Virtual OCF Server**

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

392

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

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

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

393
394
395
396

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.

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

399

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

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

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

400
401
402
403
404
405

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

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

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

406
407
408

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

409 **6.2.3.4 On-the-fly Translation**

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

412 **6.2.3.5 Protocol translation between BLE and OCF**

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

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

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

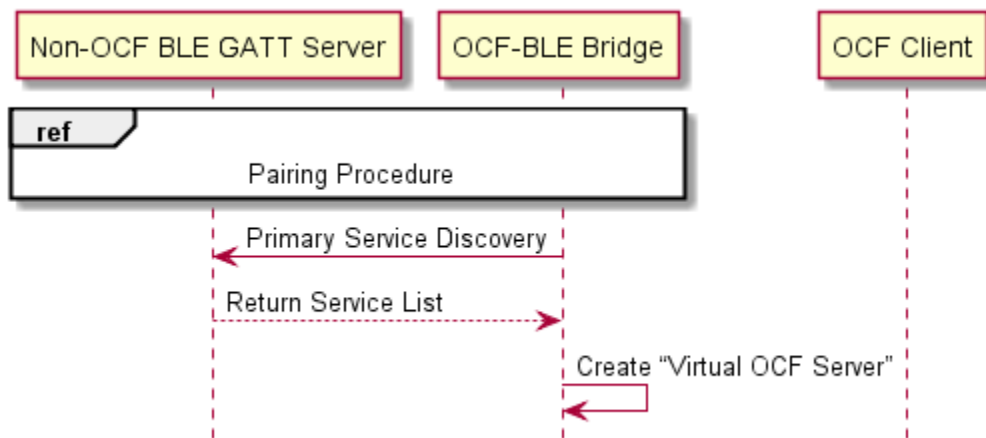
BLE GATT Feature	OCF CRUDN
N/A (Not Supported)	CREATE
Characteristic Value Read	RETRIEVE
Characteristic Value Write	UPDATE
N/A (Not Supported)	DELETE
Characteristic Descriptor Value Write	NOTIFY request
Characteristic Value Notification/Indication	NOTIFICATION response

425

426 **6.2.3.6 Illustrative OCF to BLE translation flows**

427 **6.2.3.6.1 Initialization**

428 Figure 5 shows the initial pairing procedure.



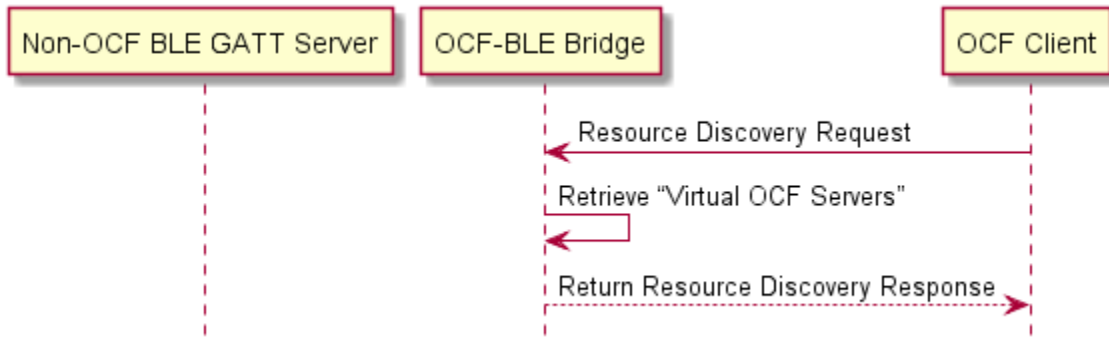
429

430

Figure 5 – Initialization

431 **6.2.3.6.2 Resource Discovery**

432 Figure 6 shows the resource discovery procedure.

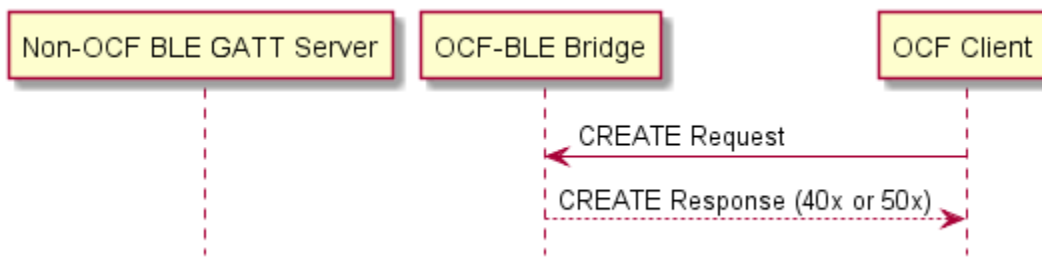


433

434 **Figure 6 – Resource Discovery**

435 **6.2.3.6.3 Create Resource**

436 Figure 7 illustrates Resource creation.

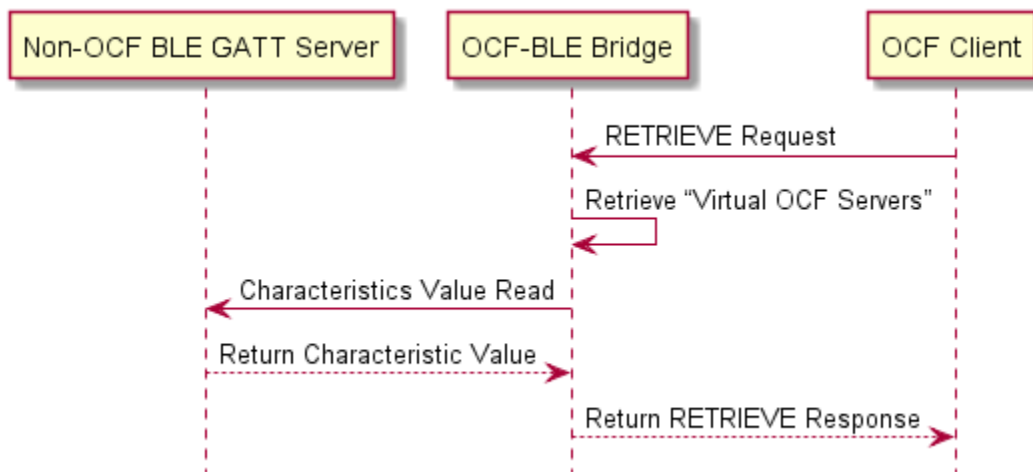


437

438 **Figure 7 – Create Resource**

439 **6.2.3.6.4 Retrieve Resource**

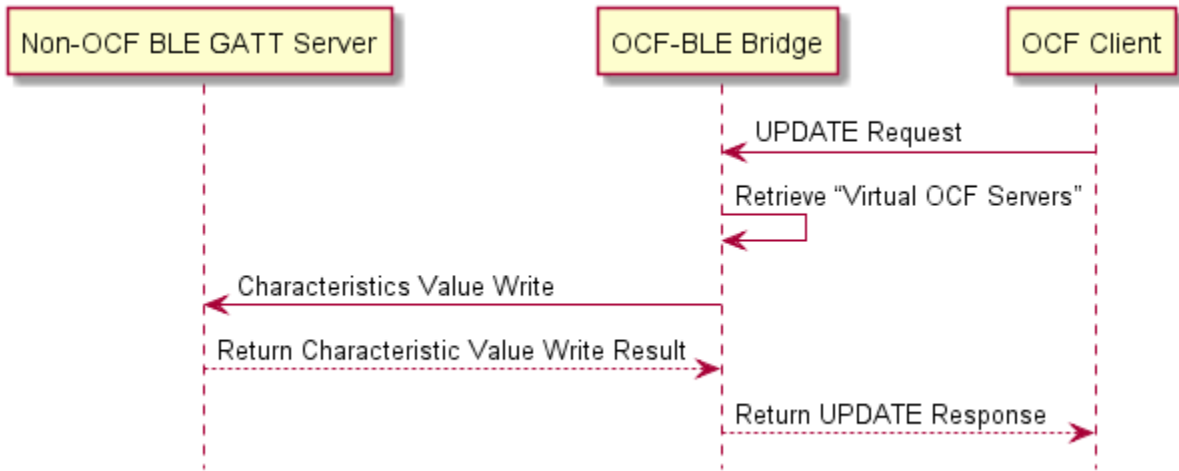
440 Figure 8 illustrates Resource RETRIEVAL.



441

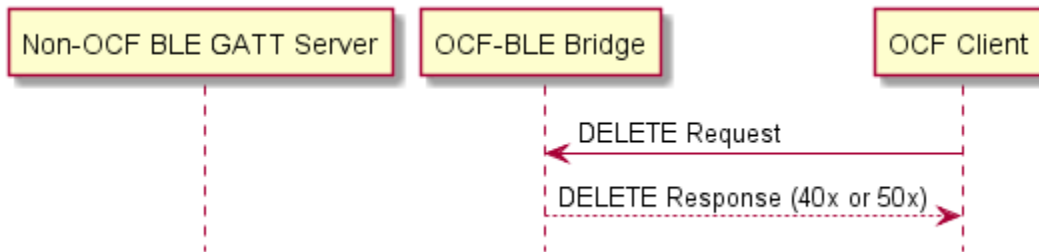
442 **Figure 8 – Retrieve Resource**

443 **6.2.3.6.5 Update Resource**
 444 Figure 9 illustrates Resource UPDATE.



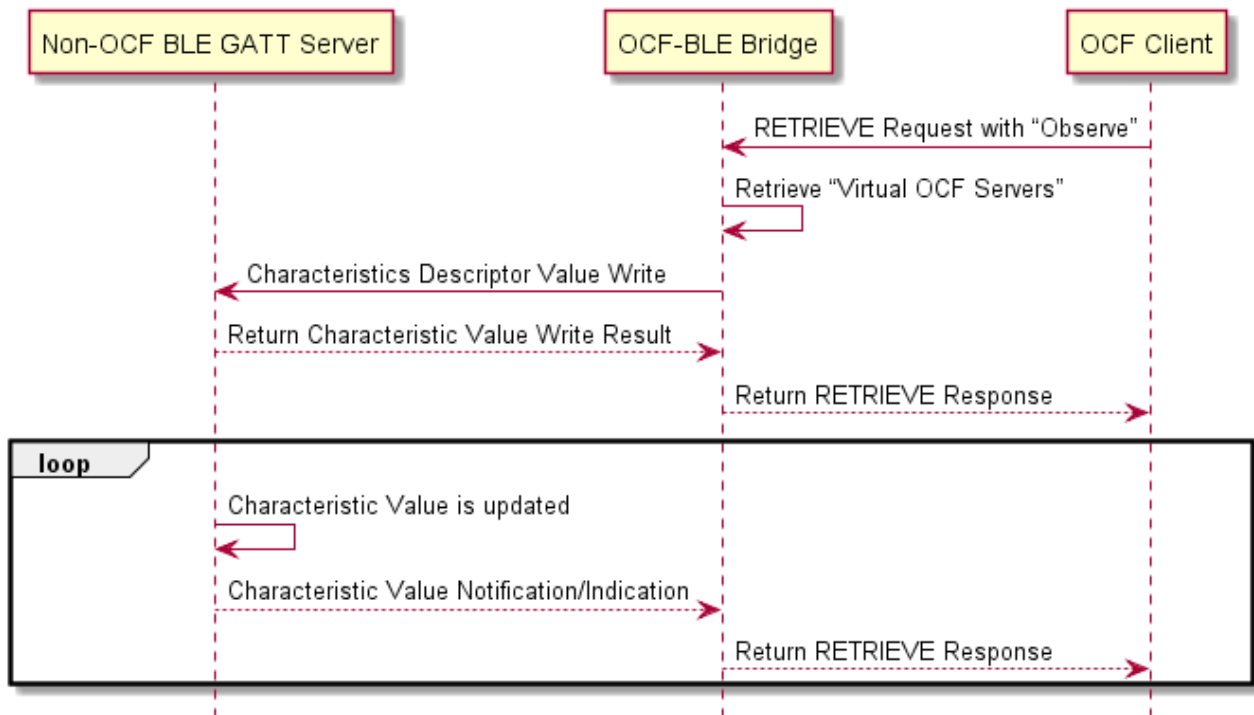
445
 446 **Figure 9 – Update Resource**

447 **6.2.3.6.6 Delete Resource**
 448 Figure 10 illustrates Resource DELETE. Note that this only applies to Resources that were created.



449
 450 **Figure 10 – Delete Resource**

451 **6.2.3.6.7 Set Notification & Send Notification**
 452 Figure 11 illustrates the establishment and sending of a notification.



453

454

Figure 11 – Set Notification and send Notification

455

6.2.3.7 Error handling

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

7 Device Type Mapping

7.1 Introduction

464 This clause contains the mappings to OCF Device Types.

7.2 BLE Profile to OCF Device Types

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

468

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"

469

470 8 BLE Profile to Resource Equivalence

471 8.1 Introduction

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

474 8.2 BLE Services to OCF Resources

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

477

Table 10 – BLE Services to OCF Resource Type Mapping

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

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

478

479 9 Detailed Mappings

480 9.1 Introduction

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

484 9.2 Blood Pressure Mapping

485 9.2.1 Derived model

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

487 9.2.2 Property definition

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

490

491

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

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

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

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

494 **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

495 9.2.3 Derived model definition

```

496 {
497   "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.BLP.json#",
498   "$schema": "http://json-schema.org/draft-04/schema#",
499   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
500 reserved.",
501   "title": "Blood Pressure",
502   "definitions": {
503     "byte": {
504       "type": "integer",
505       "minimum": 0,
506       "maximum": 255
507     },
508     "byteArray": {
509       "type": "array",
510       "items": { "$ref": "#/definitions/byte" },
511       "minItems": 1,
512       "uniqueItems": false
513     },
514     "org.bluetooth.characteristic.blood_pressure_measurement": {
515       "type": "object",
516       "properties": {
517         "blood_pressure_measurement[length - 3 : length - 1]": {
518           "$ref": "#/definitions/byteArray",
519           "description": "Blood Pressure Measurement Compound Value -
520 Systolic",
521           "x-ocf-conversion": {
522             "x-ocf-alias": "oic.r.blood.pressure",
523             "x-to-ocf": [
524               "def
525 ieee11073_Sfloat_2_Float(sfloat_value):",
526               "# reserved value for Infinity or NaN
527 (Not a Number)",
528               "reserved_float_values = {",
529                 "0x07FE:math.inf, # +INFINITY",
530                 "0x07FF:math.nan, # NaN (Not a
531 Number)",
532                 "0x0800:math.nan, # NRes (Not at this
533 Resolution)",
534                 "0x0801:math.nan, # Reserved for
535 future",
536                 "0x0802:-math.inf # -INFINITY",
537               "}",
538               "mantissa = sfloat_value & 0x0FFF",
539               "exponent = sfloat_value >> 12",
540               "if (exponent >= 0x0008):",
541                 "exponent = -((0x000F + 1) -
542 exponent)",
543               "output = 0",
544               "if (mantissa >= 0x07FE and mantissa <=
545 0x0802):",
546                 "output =
547 reserved_float_values[mantissa]",
548               "else:",
549                 "if (mantissa >= 0x0800):",
550                   "mantissa = -((0x0FFF + 1) -
551 mantissa)",
552                   "magnitude = pow(10.0, exponent)",

```

```

554                                     "output = (mantissa * magnitude)",
555                                     "return output",
556                                     "length = len(blood_pressure_measurement)",
557                                     "flags = blood_pressure_measurement[length -
558 1]",
559                                     "oic.r.blood.pressure.systolic =
560 ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 3 : length - 1])",
561                                     "oic.r.blood.pressure.units = \"mmHg\" if
562 (flags & 0x01) else \"kPa\"",
563                                     ],
564                                     "x-from-ocf": [
565                                     "N/A"
566                                     ]
567                                     },
568                                     },
569                                     "blood_pressure_measurement[length - 5 : length - 3]: {
570                                     "$ref": "#/definitions/byteArray",
571                                     "description": "Blood Pressure Measurement Compound Value -
572 Diastolic",
573                                     "x-ocf-conversion": {
574                                     "x-ocf-alias": "oic.r.blood.pressure",
575                                     "x-to-ocf": [
576                                     "def
577 ieee11073_Sfloat_2_Float(sfloat_value):",
578                                     "# reserved value for Infinity or NaN
579 (Not a Number)",
580                                     "reserved_float_values = {",
581                                     "0x07FE:math.inf, # +INFINITY",
582                                     "0x07FF:math.nan, # NaN (Not a
583 Number)",
584                                     "0x0800:math.nan, # NRes (Not at this
585 Resolution)",
586                                     "0x0801:math.nan, # Reserved for
587 future",
588                                     "0x0802:-math.inf # -INFINITY",
589                                     }",
590                                     "mantissa = sfloat_value & 0x0FFF",
591                                     "exponent = sfloat_value >> 12",
592                                     "if (exponent >= 0x0008):",
593                                     "exponent = -((0x000F + 1) -
594 exponent)",
595                                     "output = 0",
596                                     "if (mantissa >= 0x07FE and mantissa <=
597 0x0802):",
598                                     "output =
599 reserved_float_values[mantissa]",
600                                     "else:",
601                                     "if (mantissa >= 0x0800):",
602                                     "mantissa = -((0x0FFF + 1) -
603 mantissa)",
604                                     "magnitude = pow(10.0, exponent)",
605                                     "output = (mantissa * magnitude)",
606                                     "return output",
607                                     "length = len(blood_pressure_measurement)",
608                                     "oic.r.blood.pressure.diastolic =
609 ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 5 : length - 3])"
610                                     ],
611                                     "x-from-ocf": [
612                                     "N/A"
613                                     ]
614                                     },
615                                     },
616                                     "blood_pressure_measurement[length - 7 : length - 5]: {
617                                     "$ref": "#/definitions/byteArray",
618                                     "description": "Blood Pressure Measurement Compound Value -
619 Mean Arterial Pressure",
620                                     "x-ocf-conversion": {
621                                     "x-ocf-alias": "oic.r.blood.pressure",
622                                     "x-to-ocf": [
623                                     "def
624 ieee11073_Sfloat_2_Float(sfloat_value):",

```

```

625                                     "# reserved value for Infinity or NaN
626 (Not a Number)",
627                                     "reserved_float_values = {",
628                                     "0x07FE:math.inf, # +INFINITY",
629                                     "0x07FF:math.nan, # NaN (Not a
630 Number)",
631                                     "0x0800:math.nan, # NRes (Not at this
632 Resolution)",
633                                     "0x0801:math.nan, # Reserved for
634 future",
635                                     "0x0802:-math.inf # -INFINITY",
636                                     "}",
637                                     "mantissa = sfloat_value & 0x0FFF",
638                                     "exponent = sfloat_value >> 12",
639                                     "if (exponent >= 0x0008):",
640                                     "exponent = -((0x000F + 1) -
641 exponent)",
642                                     "output = 0",
643                                     "if (mantissa >= 0x07FE and mantissa <=
644 0x0802):",
645                                     "output =
646 reserved_float_values[mantissa]",
647                                     "else:",
648                                     "if (mantissa >= 0x0800):",
649                                     "mantissa = -((0x0FFF + 1) -
650 mantissa)",
651                                     "magnitude = pow(10.0, exponent)",
652                                     "output = (mantissa * magnitude)",
653                                     "return output",
654                                     "length = len(blood_pressure_measurement)",
655                                     "oic.r.blood.pressure.map =
656 ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 7 : length - 5])"
657                                     ],
658                                     "x-from-ocf": [
659                                     "N/A"
660                                     ]
661                                     }
662                                     },
663                                     "blood_pressure_measurement[length - 7 - timestamp_len - 2 : length
664 - 7 - timestamp_len]": {
665                                     "$ref": "#/definitions/byteArray",
666                                     "description": "Pulse Rate",
667                                     "x-ocf-conversion": {
668                                     "x-ocf-alias": "oic.r.pulserate",
669                                     "x-to-ocf": [
670                                     "def
671 ieee11073_Sfloat_2_Float(sfloat_value):",
672                                     "# reserved value for Infinity or NaN
673 (Not a Number)",
674                                     "reserved_float_values = {",
675                                     "0x07FE:math.inf, # +INFINITY",
676                                     "0x07FF:math.nan, # NaN (Not a
677 Number)",
678                                     "0x0800:math.nan, # NRes (Not at this
679 Resolution)",
680                                     "0x0801:math.nan, # Reserved for
681 future",
682                                     "0x0802:-math.inf # -INFINITY",
683                                     "}",
684                                     "mantissa = sfloat_value & 0x0FFF",
685                                     "exponent = sfloat_value >> 12",
686                                     "if (exponent >= 0x0008):",
687                                     "exponent = -((0x000F + 1) -
688 exponent)",
689                                     "output = 0",
690                                     "if (mantissa >= 0x07FE and mantissa <=
691 0x0802):",
692                                     "output =
693 reserved_float_values[mantissa]",
694                                     "else:",
695                                     "if (mantissa >= 0x0800):",

```

```

696         "mantissa = -((0x0FFF + 1) -
697 mantissa)",
698         "magnitude = pow(10.0, exponent)",
699         "output = (mantissa * magnitude)",
700         "return output",
701         "length = len(blood_pressure_measurement)",
702         "flags = blood_pressure_measurement[length -
703 1]",
704         "timestamp_len = 7 if (flags & 0x02) else 0",
705         "oic.r.pulserate.pulserate =
706 ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 7 - timestamp_len - 2 : length - 7 -
707 timestamp_len])"
708     ],
709     "x-from-ocf": [
710         "N/A"
711     ]
712 }
713 }
714 }
715 }
716 },
717 "type": "object",
718 "allOf": [
719     { "$ref": "#/definitions/byte" },
720     { "$ref": "#/definitions/byteArray" },
721     { "$ref": "#/definitions/org.bluetooth.characteristic.blood_pressure_measurement" }
722 ],
723 "required": [
724     "blood_pressure_measurement[length - 3 : length - 1]",
725     "blood_pressure_measurement[length - 5 : length - 3]"
726 ]
727 }
728 }
729 }
730 }
731 }

```

9.3 Glucose Measurement Mapping

9.3.1 Derived model

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

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

9.3.2 Property definition

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

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

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

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

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

742 **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

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

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

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

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

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

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

749
750

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

751 **9.3.3 Derived model definition**

```

752 {
753   "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.GLP.json#",
754   "$schema": "http://json-schema.org/draft-04/schema#",
755   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
756 reserved.",
757   "title": "Glucose",
758   "definitions": {
759     "byte": {
760       "type": "integer",
761       "minimum": 0,
762       "maximum": 255
763     },
764     "byteArray": {
765       "type": "array",
766       "items": { "$ref": "#/definitions/byte" },
767       "minItems": 1,
768       "uniqueItems": false
769     },
770     "org.bluetooth.characteristic.glucose_measurement": {
771       "type": "object",
772       "properties": {
773         "glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len -
774 10]": {
775           "$ref": "#/definitions/byteArray",
776           "description": "Glucose Concentration",
777           "x-ocf-conversion": {
778             "x-ocf-alias": "oic.r.glucose",
779             "x-to-ocf": [
780               "def ieee11073_Sfloat_2_Float(sfloat_value)",
781               "# reserved value for Infinity or NaN (Not a Number)",
782               "reserved_float_values = {",
783                 "0x07FE:math.inf, # +INFINITY",
784                 "0x07FF:math.nan, # NaN (Not a Number)",
785                 "0x0800:math.nan, # NRes (Not at this Resolution)",

```

```

787         "0x0801:math.nan, # Reserved for future",
788         "0x0802:-math.inf # -INFINITY",
789     "}",
790     "mantissa = sfloat_value & 0x0FFF",
791     "exponent = sfloat_value >> 12",
792     "if (exponent >= 0x0008):",
793         "exponent = -((0x000F + 1) - exponent)",
794     "output = 0",
795     "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
796         "output = reserved_float_values[mantissa]",
797     "else:",
798         "if (mantissa >= 0x0800):",
799             "mantissa = -((0x0FFF + 1) - mantissa)",
800             "magnitude = pow(10.0, exponent)",
801             "output = (mantissa * magnitude)",
802         "return output",
803     "length = len(glucose_measurement)",
804     "flags = glucose_measurement[length - 1]",
805     "timeoffset_len = 2 if (flags & 0x01) else 0",
806     "if (flags & 0x02) == True:",
807         " glucose = ieee11073_Sfloat_2_Float(glucose_measurement[length - 2 -
808 timeoffset_len - 10 : length - timeoffset_len - 10])",
809         " oic.r.glucose.glucose = (glucose * 1000) if (flags & 0x04) else
810 (glucose * 0.1 * 1000 * 1000)",
811         " oic.r.glucose.units = 'mmol/L' if (flags & 0x04) else 'mg/dL'",
812     "if (flags & 0x02) == False:",
813         " oic.r.glucose.glucose = 0",
814         " oic.r.glucose.units = 'mmol/L'"
815     ],
816     "x-from-ocf": [
817         "N/A"
818     ]
819 }
820 },
821 "glucose_measurement[length - 1 - 2 - timeoffset_len - 10]": {
822     "$ref": "#/definitions/byteArray",
823     "description": "Sample Location",
824     "x-ocf-conversion": {
825         "x-ocf-alias": "oic.r.glucose.samplelocation",
826         "x-to-ocf": [
827             "length = len(glucose_measurement)",
828             "flags = glucose_measurement[length - 1]",
829             "timeoffset_len = 2 if (flags & 0x01) else 0",
830             "if (flags & 0x02):",
831                 " samplelocation = { 1:'finger', 2:'ast', 3:'earlobe',
832 4:'ctrlsolution' }",
833             " oic.r.glucose.samplelocation.samplelocation =
834 samplelocation[glucose_measurement[length - 1 - 2 - timeoffset_len - 10] & 0xf0]"
835         ],
836         "x-from-ocf": [
837             "N/A"
838         ]
839     }
840 }
841 }
842 },
843
844 "org.bluetooth.characteristic.glucose_measurement_context": {
845     "type": "object",
846     "properties": {
847         "glucose_measurement_context[length - carb_len - extflags_len - 3 : length - 1 -
848 extflags_len - 3]": {
849             "$ref": "#/definitions/byteArray",
850             "description": "Carbohydrate",
851             "x-ocf-conversion": {
852                 "x-ocf-alias": "oic.r.glucose.carb",
853                 "x-to-ocf": [
854                     "def ieee11073_Sfloat_2_Float(sfloat_value):",
855                     "# reserved value for Infinity or NaN (Not a Number)",
856                     "reserved_float_values = {",
857                     "0x07FE:math.inf, # +INFINITY",

```

```

858         "0x07FF:math.nan, # NaN (Not a Number)",
859         "0x0800:math.nan, # NRes (Not at this Resolution)",
860         "0x0801:math.nan, # Reserved for future",
861         "0x0802:-math.inf # -INFINITY",
862     },
863     "mantissa = sfloat_value & 0x0FFF",
864     "exponent = sfloat_value >> 12",
865     "if (exponent >= 0x0008):",
866         "exponent = -((0x000F + 1) - exponent)",
867     "output = 0",
868     "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
869         "output = reserved_float_values[mantissa]",
870     "else:",
871         "if (mantissa >= 0x0800):",
872             "mantissa = -((0x0FFF + 1) - mantissa)",
873             "magnitude = pow(10.0, exponent)",
874             "output = (mantissa * magnitude)",
875         "return output",
876     "length = len(glucose_measurement_context)",
877     "flags = glucose_measurement_context[length - 1]",
878     "extflags_len = 1 if (flags & 0x80) else 0",
879     "carb_len = 3 if (flags & 0x01) else 0",
880     "if (flags & 0x01): ",
881     "    oic.r.glucose.carb.carb =
882 ieee11073_Sfloat_2_Float(glucose_measurement_context[length - carb_len - extflags_len - 3 : length
883 - 1 - extflags_len - 3]) * 1000"
884     ],
885     "x-from-ocf": [
886         "N/A"
887     ]
888     }
889 },
890 "glucose_measurement_context[length - 1 - extflags_len - 3]": {
891     "$ref": "#/definitions/byteArray",
892     "description": "Carbohydrate ID",
893     "x-ocf-conversion": {
894         "x-ocf-alias": "oic.r.glucose.carb",
895         "x-to-ocf": [
896             "length = len(glucose_measurement_context)",
897             "flags = glucose_measurement_context[length - 1]",
898             "extflags_len = 1 if (flags & 0x80) else 0",
899             "if (flags & 0x01): ",
900             "    meal = { 1:'breakfast', 2:'lunch', 3:'dinner', 4:'snack',
901 5:'drink', 6:'supper', 7:'brunch' }",
902             "    oic.r.glucose.carb.meal = meal[glucose_measurement_context[length -
903 1 - extflags_len - 3]]"
904         ],
905         "x-from-ocf": [
906             "N/A"
907         ]
908     }
909 },
910 "glucose_measurement_context[length - 2 - health_len - meal_len - carb_len -
911 extflags_len - 3]": {
912     "$ref": "#/definitions/byteArray",
913     "description": "Exercise Intensity",
914     "x-ocf-conversion": {
915         "x-ocf-alias": "oic.r.glucose.exercise",
916         "x-to-ocf": [
917             "length = len(glucose_measurement_context)",
918             "flags = glucose_measurement_context[length - 1]",
919             "extflags_len = 1 if (flags & 0x80) else 0",
920             "carb_len = 3 if (flags & 0x01) else 0",
921             "meal_len = 1 if (flags & 0x02) else 0",
922             "health_len = 1 if (flags & 0x04) else 0",
923             "if (flags & 0x08): ",
924             "    oic.r.glucose.exercise.exercise =
925 glucose_measurement_context[length - 2 - health_len - meal_len - carb_len - extflags_len - 3]"
926         ],
927         "x-from-ocf": [
928             "N/A"

```

```

929         ]
930     }
931 },
932     "glucose_measurement_context[length - hbalc_len - medication_len - exercise_len -
933 health_len - meal_len - carb_len - extflags_len - 3 : length - medication_len - exercise_len -
934 health_len - meal_len - carb_len - extflags_len - 3]": {
935     "$ref": "#/definitions/byteArray",
936     "description": "HbA1c",
937     "x-ocf-conversion": {
938         "x-ocf-alias": "oic.r.glucose.hbalc",
939         "x-to-ocf": [
940             "def ieeel1073_Sfloat_2_Float(sfloat_value):",
941             "# reserved value for Infinity or NaN (Not a Number)",
942             "reserved_float_values = {",
943             "    0x07FE:math.inf, # +INFINITY",
944             "    0x07FF:math.nan, # NaN (Not a Number)",
945             "    0x0800:math.nan, # NRes (Not at this Resolution)",
946             "    0x0801:math.nan, # Reserved for future",
947             "    0x0802:-math.inf # -INFINITY",
948             "}",
949             "mantissa = sfloat_value & 0x0FFF",
950             "exponent = sfloat_value >> 12",
951             "if (exponent >= 0x0008):",
952                 "    exponent = -((0x000F + 1) - exponent)",
953             "output = 0",
954             "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
955                 "    output = reserved_float_values[mantissa]",
956             "else:",
957                 "if (mantissa >= 0x0800):",
958                     "    mantissa = -((0x0FFF + 1) - mantissa)",
959                     "magnitude = pow(10.0, exponent)",
960                     "output = (mantissa * magnitude)",
961             "return output",
962             "length = len(glucose_measurement_context)",
963             "flags = glucose_measurement_context[length - 1]",
964             "extflags_len = 1 if (flags & 0x80) else 0",
965             "carb_len = 3 if (flags & 0x01) else 0",
966             "meal_len = 1 if (flags & 0x02) else 0",
967             "health_len = 1 if (flags & 0x04) else 0",
968             "exercise_len = 3 if (flags & 0x08) else 0",
969             "medication_len = 3 if (flags & 0x10) else 0",
970             "hbalc_len = 2 if (flags & 0x40) else 0",
971             "if (flags & 0x40):",
972                 "    oic.r.glucose.hbalc.hbalc =
973 ieeel1073_Sfloat_2_Float(glucose_measurement_context[length - hbalc_len - medication_len -
974 exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - medication_len -
975 exercise_len - health_len - meal_len - carb_len - extflags_len - 3])"
976         ],
977         "x-from-ocf": [
978             "N/A"
979         ]
980     ]
981 },
982     "glucose_measurement_context[length - health_len - meal_len - carb_len -
983 extflags_len - 3]": {
984     "$ref": "#/definitions/byteArray",
985     "description": "Health, Tester",
986     "x-ocf-conversion": {
987         "x-ocf-alias": "oic.r.glucose.health",
988         "x-to-ocf": [
989             "length = len(glucose_measurement_context)",
990             "flags = glucose_measurement_context[length - 1]",
991             "extflags_len = 1 if (flags & 0x80) else 0",
992             "carb_len = 3 if (flags & 0x01) else 0",
993             "meal_len = 1 if (flags & 0x02) else 0",
994             "health_len = 1 if (flags & 0x04) else 0",
995             "if (flags & 0x04):",
996                 "    health = { 1:'minor', 2:'major', 3:'menses', 4:'stress',
997 5:'none' }",
998             "    oic.r.glucose.health.health =
999 health[glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3] &

```

```

1000 0xf0]",
1001         " tester = { 1:'self', 2:'hcp', 3:'lab' }",
1002         " oic.r.glucose.tester.tester =
1003 tester[glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3] &
1004 0xf0]"
1005     ],
1006     "x-from-ocf": [
1007         "N/A"
1008     ]
1009 },
1010 },
1011 "glucose_measurement_context[length - meal_len - carb_len - extflags_len - 3]: {
1012     "$ref": "#/definitions/byteArray",
1013     "description": "Meal",
1014     "x-ocf-conversion": {
1015         "x-ocf-alias": "oic.r.glucose.meal",
1016         "x-to-ocf": [
1017             "length = len(glucose_measurement_context)",
1018             "flags = glucose_measurement_context[length - 1]",
1019             "extflags_len = 1 if (flags & 0x80) else 0",
1020             "carb_len = 3 if (flags & 0x01) else 0",
1021             "meal_len = 1 if (flags & 0x02) else 0",
1022             "if (flags & 0x02): ",
1023             " meal = { 1:'preprandial', 2:'postprandial', 3:'fasting',
1024 4:'casual', 5:'bedtime' }",
1025             " oic.r.glucose.meal.meal = meal[glucose_measurement_context[length -
1026 meal_len - carb_len - extflags_len - 3]]"
1027         ],
1028         "x-from-ocf": [
1029             "N/A"
1030         ]
1031     }
1032 },
1033 "glucose_measurement_context[length - medication_len - exercise_len - health_len -
1034 meal_len - carb_len - extflags_len - 3 : length - 1 - exercise_len - health_len - meal_len -
1035 carb_len - extflags_len - 3]: {
1036     "$ref": "#/definitions/byteArray",
1037     "description": "Medication",
1038     "x-ocf-conversion": {
1039         "x-ocf-alias": "oic.r.glucose.medication",
1040         "x-to-ocf": [
1041             "def ieee11073_Sfloat_2_Float(sfloat_value):",
1042             "# reserved value for Infinity or NaN (Not a Number)",
1043             "reserved_float_values = {",
1044             " 0x07FE:math.inf, # +INFINITY",
1045             " 0x07FF:math.nan, # NaN (Not a Number)",
1046             " 0x0800:math.nan, # NRes (Not at this Resolution)",
1047             " 0x0801:math.nan, # Reserved for future",
1048             " 0x0802:-math.inf # -INFINITY",
1049             "}",
1050             "mantissa = sfloat_value & 0x0FFF",
1051             "exponent = sfloat_value >> 12",
1052             "if (exponent >= 0x0008):",
1053             "   exponent = -((0x000F + 1) - exponent)",
1054             "output = 0",
1055             "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
1056             "   output = reserved_float_values[mantissa]",
1057             "else:",
1058             "   if (mantissa >= 0x0800):",
1059             "       mantissa = -((0x0FFF + 1) - mantissa)",
1060             "   magnitude = pow(10.0, exponent)",
1061             "   output = (mantissa * magnitude)",
1062             "return output",
1063             "length = len(glucose_measurement_context)",
1064             "flags = glucose_measurement_context[length - 1]",
1065             "extflags_len = 1 if (flags & 0x80) else 0",
1066             "carb_len = 3 if (flags & 0x01) else 0",
1067             "meal_len = 1 if (flags & 0x02) else 0",
1068             "health_len = 1 if (flags & 0x04) else 0",
1069             "exercise_len = 3 if (flags & 0x08) else 0",
1070             "medication_len = 3 if (flags & 0x10) else 0",

```

```

1071         "hbalc_len = 2 if (flags & 0x40) else 0",
1072         "if (flags & 0x10): ",
1073         "     medication =
1074 iieee11073_Sfloat_2_Float(glucose_measurement_context[length - medication_len - exercise_len -
1075 health_len - meal_len - carb_len - extflags_len - 3 : length - 1 - exercise_len - health_len -
1076 meal_len - carb_len - extflags_len - 3])",
1077         "     oic.r.glucose.medication.medication = medication * 1000",
1078         "     oic.r.glucose.medication.units = 'mL' if (flags & 0x20) else 'mg'"
1079     ],
1080     "x-from-ocf": [
1081         "N/A"
1082     ]
1083     }
1084 },
1085     "glucose_measurement_context[length - 1 - exercise_len - health_len - meal_len -
1086 carb_len - extflags_len - 3]": {
1087         "$ref": "#/definitions/byteArray",
1088         "description": "Medication ID",
1089         "x-ocf-conversion": {
1090             "x-ocf-alias": "oic.r.glucose.medication",
1091             "x-to-ocf": [
1092                 "length = len(glucose_measurement_context)",
1093                 "flags = glucose_measurement_context[length - 1]",
1094                 "extflags_len = 1 if (flags & 0x80) else 0",
1095                 "carb_len = 3 if (flags & 0x01) else 0",
1096                 "meal_len = 1 if (flags & 0x02) else 0",
1097                 "health_len = 1 if (flags & 0x04) else 0",
1098                 "exercise_len = 3 if (flags & 0x08) else 0",
1099                 "if (flags & 0x10): ",
1100                 "     regimen = { 1:'rapidacting', 2:'shortacting',
1101 3:'intermediateacting', 4:'longacting', 5:'premix' }",
1102                 "     oic.r.glucose.medication.regimen =
1103 regimen[ glucose_measurement_context[length - 1 - exercise_len - health_len - meal_len - carb_len -
1104 extflags_len - 3] ]"
1105             ],
1106             "x-from-ocf": [
1107                 "N/A"
1108             ]
1109         }
1110     }
1111 }
1112 }
1113 },
1114     "type": "object",
1115     "allOf": [
1116         { "$ref": "#/definitions/byte" },
1117         { "$ref": "#/definitions/byteArray" },
1118         { "$ref":
1119 "#/definitions/org.bluetooth.characteristic.org.bluetooth.characteristic.glucose_measurement" },
1120         { "$ref":
1121 "#/definitions/org.bluetooth.characteristic.org.bluetooth.characteristic.glucose_measurement_context" }
1122     ],
1123     "required": [
1124         "glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len -
1125 10]"
1126     ]
1127 }
1128 }
1129 }
1130 }
1131 }
1132 }

```

1133 9.4 Health Thermometer Mapping

1134 9.4.1 Derived model

1135 The derived model: "org.bluetooth.characteristic.temperature_measurement".

1136 **9.4.2 Property definition**

1137 Table 17 provides the detailed per Property mapping for
 1138 "org.bluetooth.characteristic.temperature_measurement".

1139 **Table 17 – The Property mapping for**
 1140 **"org.bluetooth.characteristic.temperature_measurement".**

BLE Property name	OCF Resource	To OCF	From OCF
temperature_measurement[length - 5 : length - 1]	oic.r.temperature	# convert IEEE11073 FLOAT to floatdef ieee11073_Float_2_Float(float_value):# reserved value for Infinity or NaN (Not a Number)reserved_float_values = {0x007FFFFE:math.inf, # +INFINITY0x007FFFFF:math.nan, # NaN (Not a Number)0x00800000:math.nan, # NRes (Not at this Resolution)0x00800001:math.nan, # Reserved for future0x00800002:-math.inf # - INFINITY}mantissa = float_value & 0x00FFFFFFexponent = float_value >> 24if (exponent >= 0x00000080):exponent = - ((0x000000FF + 1) - exponent)output = 0if (mantissa >= 0x007FFFFE and mantissa <= 0x00800002):output = reserved_float_values[mantissa]else:if (mantissa >= 0x00800000):mantissa = - ((0x00FFFFFF + 1) - mantissa)magnitude = pow(10.0, exponent)output = (mantissa * magnitude)return outputlength = len(temperature_measurement)flags = temperature_measurement[length - 1]oic.r.temperature.temperature = ieee11073_Float_2_Float(temperature_measur e ment[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.tempe rature	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

1141 Table 18 provides the details of the Properties that are part of
 1142 "org.bluetooth.characteristic.temperature_measurement".

1143 **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

1144 9.4.3 Derived model definition

```
1145 {
1146   "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.HTP.json#",
1147   "$schema": "http://json-schema.org/draft-04/schema#",
1148   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
1149 reserved.",
1150   "title": "Health Thermometer",
1151   "definitions": {
1152     "byte": {
1153       "type": "integer",
1154       "minimum": 0,
1155       "maximum": 255
1156     },
1157     "byteArray": {
1158       "type": "array",
1159       "items": { "$ref": "#/definitions/byte" },
1160       "minItems": 1,
1161       "uniqueItems": false
1162     },
1163     "org.bluetooth.characteristic.temperature_measurement" : {
1164       "type": "object",
1165       "properties" : {
1166         "temperature_measurement[ length - 5 : length - 1 ]" : {
1167           "$ref": "#/definitions/byteArray",
1168           "description": "Temperature",
1169           "x-ocf-conversion": {
1170             "x-ocf-alias": "oic.r.temperature",
1171             "x-to-ocf": [
1172               "# convert IEEE11073 FLOAT to float",
1173               "def ieee11073_Float_2_Float(float_value):",
1174                 "# reserved value for Infinity or NaN
1175               "reserved_float_values = {",
1176                 "0x007FFFFE:math.inf, # +INFINITY",
1177                 "0x007FFFFFF:math.nan, # NaN (Not a
1178                 Number)",
1179                 "0x00800000:math.nan, # NRes (Not at
1180                 this Resolution)",
1181                 "0x00800001:math.nan, # Reserved for
1182                 future",
1183                 "0x00800002:-math.inf # -INFINITY",
1184               "}",
1185               "mantissa = float_value & 0x00FFFFFF",
1186               "exponent = float_value >> 24",
1187               "if (exponent >= 0x00000080):",
1188                 "exponent = -((0x000000FF + 1) -
1189                 exponent)",
1190               "output = 0",
1191               "if (mantissa >= 0x007FFFFE and mantissa
1192               <= 0x00800002):",
1193                 "output =
1194                 reserved_float_values[mantissa]",
1195               "else:",
1196                 "if (mantissa >= 0x00800000):",
1197                   "mantissa = -((0x00FFFFFF + 1) -
1198                   mantissa)",
1199                   "magnitude = pow(10.0, exponent)",
1200                   "output = (mantissa * magnitude)",
1201                   "return output",
1202               "length = len(temperature_measurement)",
1203               "flags = temperature_measurement[length -
1204               1]",
1205               "oic.r.temperature.temperature =
1206               ieee11073_Float_2_Float(temperature_measurement[ length - 5 : length - 1 ])",
1207               "oic.r.temperature.units = 'F' if (flags &
1208               0x01) else 'C'
1209             ]
1210           },
1211           "x-from-ocf": [
1212             "N/A"
1213           ]
1214         }
1215       }
1216     }
1217   }
1218 }
```

```

1214         ]
1215     }
1216 },
1217     "temperature_measurement[ length - temperaturetype_len -
1218 timestamp_len - 5 ]" : {
1219         "$ref": "#/definitions/byteArray",
1220         "description": "Temperature Type",
1221         "x-ocf-conversion": {
1222             "x-ocf-alias": "oic.r.body.location.temperature",
1223             "x-to-ocf": [
1224                 "length = len(temperature_measurement)",
1225                 "flags = temperature_measurement[length -
1226 1]",
1227                 "timestamp_len = 7 if (flags & 0x02) 0",
1228                 "temperaturetype_len = 1 if (flags & 0x04)
1229 0",
1230                 "if (flags & 0x04):",
1231                 "    bloc = { 1:'xxx', 2:'body', 3:'ear',
1232 4:'finger', 5:'gastro', 6:'mouth', 7:'rectum', 8:'toe', 9:'tympaanum' }",
1233                 "    oic.r.body.location.temperature.bloc =
1234 bloc[temperature_measurement[ length - temperaturetype_len - timestamp_len - 5 ]]"
1235             ],
1236             "x-from-ocf": [
1237                 "N/A"
1238             ]
1239         }
1240     }
1241 }
1242 },
1243 },
1244 },
1245 "type": "object",
1246 "allOf": [
1247     { "$ref": "#/definitions/byte" },
1248     { "$ref": "#/definitions/byteArray" },
1249     { "$ref": "#/definitions/org.bluetooth.characteristic.temperature_measurement" }
1250 ],
1251 ],
1252 "required": [
1253     "temperature_measurement[ length - 5 : length - 1 ]"
1254 ]
1255 }
1256 }
1257

```

1258 9.5 Weight Scale Mapping

1259 9.5.1 Derived model

1260 The derived model: "org.bluetooth.characteristic.weight_measurement".

1261 The derived model: "org.bluetooth.characteristic.body_composition_measurement".

1262 9.5.2 Property definition

1263 Table 19 provides the detailed per Property mapping for
1264 "org.bluetooth.characteristic.weight_measurement".

1265 **Table 19 – The Property mapping for "org.bluetooth.characteristic.weight_measurement".**

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

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

1266 Table 20 provides the details of the Properties that are part of
1267 "org.bluetooth.characteristic.weight_measurement".

1268 **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

1269 Table 21 provides the detailed per Property mapping for
1270 "org.bluetooth.characteristic.body_composition_measurement".

1271 **Table 21 – The Property mapping for**
1272 **"org.bluetooth.characteristic.body_composition_measurement".**

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

		<pre> Offm_len = 2 if (flags_lowerbyte & 0x40) else Oslm_len = 2 if (flags_lowerbyte & 0x80) else Obwm_len = 2 if (flags_upperbyte & 0x01) else 0if (flags_lowerbyte & 0x01): oic.r.body.water.bwater = int.from_bytes(body_composition_measurement[length - bwm_len - slm_len - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4], 'big') oic.r.body.water.units = 'lb' if (flags_lowerbyte & 0x01) 'kg' </pre>	
<pre> 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] </pre>	oic.r.body.slm	<pre> length = len(body_composition_measurement)flags_upperbyte 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[length - slm_len - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4], 'big') oic.r.body.slm.units = 'lb' if (flags_lowerbyte & 0x01) 'kg' </pre>	N/A
<pre> 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] </pre>	oic.r.body.ffm	<pre> length = len(body_composition_measurement)flags_upperbyte 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[length - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4], 'big') oic.r.body.ffm.units = 'lb' if (flags_lowerbyte & 0x01) 'kg' </pre>	N/A

1273 Table 22 provides the details of the Properties that are part of
1274 "org.bluetooth.characteristic.body_composition_measurement".

1275 **Table 22 – The Properties of**
1276 **"org.bluetooth.characteristic.body_composition_measurement".**

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

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

1277 **9.5.3 Derived model definition**

```

1278 {
1279     "id": "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.WSS.json#",
1280     "$schema": "http://json-schema.org/draft-04/schema#",
1281     "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
1282 reserved.",
1283     "title": "Weight Scale",
1284     "definitions": {
1285         "byte": {
1286             "type": "integer",
1287             "minimum": 0,
1288             "maximum": 255
1289         },
1290         "byteArray": {
1291             "type": "array",
1292             "items": { "$ref": "#/definitions/byte" },
1293             "minItems": 1,
1294             "uniqueItems": false
1295         },
1296         "org.bluetooth.characteristic.weight_measurement" : {
1297             "type": "object",
1298             "properties": {
1299                 "weight_measurement[length - 3 : length - 1]" : {
1300                     "$ref": "#/definitions/byteArray",
1301                     "description": "Weight",
1302                     "x-ocf-conversion": {
1303                         "x-ocf-alias": "oic.r.weight",
1304                         "x-to-ocf": [
1305                             "length = len(weight_measurement)",
1306                             "flags = weight_measurement[length - 1]",
1307                             "timeoffset_len = 7 if (flags & 0x02) else
1308 0",
1309                             "oic.r.weight.weight =
1310 int.from_bytes(weight_measurement[length - 3 : length - 1], 'big')",
1311                             "oic.r.weight.units = 'lb' if (flags & 0x01)
1312 else 'kg'"
1313                         ],
1314                         "x-from-ocf": [
1315                             "N/A"
1316                         ]
1317                     }
1318                 },
1319                 "weight_measurement[length - 2 - userid_len - timeoffset_len - 3 :
1320 length - userid_len - timeoffset_len - 3]" : {
1321                     "$ref": "#/definitions/byteArray",
1322                     "description": "BMI",
1323                     "x-ocf-conversion": {
1324                         "x-ocf-alias": "oic.r.bmi",
1325                         "x-to-ocf": [
1326                             "length = len(weight_measurement)",
1327                             "flags = weight_measurement[length - 1]",
1328                             "timeoffset_len = 7 if (flags & 0x02) else
1329 0",
1330

```

```

1331                                     "userid_len = 1 if (flags & 0x04) else 0",
1332                                     "if (flags & 0x08):",
1333                                     "    oic.r.bmi.bmi =
1334 int.from_bytes(weight_measurement[length - 2 - userid_len - timeoffset_len - 3 : length -
1335 userid_len - timeoffset_len - 3], 'big')"
1336                                     ],
1337                                     "x-from-ocf": [
1338                                         "N/A"
1339                                     ]
1340                                 }
1341                             },
1342                             "weight_measurement[length - height_len - userid_len -
1343 timeoffset_len - 3 : length - 2 - userid_len - timeoffset_len - 3]" : {
1344                                 "$ref": "#/definitions/byteArray",
1345                                 "description": "Height",
1346                                 "x-ocf-conversion": {
1347                                     "x-ocf-alias": "oic.r.height",
1348                                     "x-to-ocf": [
1349                                         "length = len(weight_measurement)",
1350                                         "flags = weight_measurement[length - 1]",
1351                                         "timeoffset_len = 7 if (flags & 0x02) else
1352 0",
1353                                         "userid_len = 1 if (flags & 0x04) else 0",
1354                                         "height_len = 4 if (flags & 0x08) else 0",
1355                                         "if (flags & 0x08):",
1356                                         "    oic.r.height.height =
1357 int.from_bytes(weight_measurement[length - height_len - userid_len - timeoffset_len - 3 : length -
1358 2 - userid_len - timeoffset_len - 3], 'big')",
1359                                         "    oic.r.height.units = 'in' if (flags &
1360 0x01) else 'm'"
1361                                     ],
1362                                     "x-from-ocf": [
1363                                         "N/A"
1364                                     ]
1365                                 }
1366                             }
1367                         },
1368                     },
1369                 "org.bluetooth.characteristic.body_composition_measurement" : {
1370                     "type": "object",
1371                     "properties": {
1372                         "body_composition_measurement[ length - 4 : length - 2]" : {
1373                             "$ref": "#/definitions/byteArray",
1374                             "description": "Body Fat Percentage",
1375                             "x-ocf-conversion": {
1376                                 "x-ocf-alias": "oic.r.body.fat",
1377                                 "x-to-ocf": [
1378                                     "length = len(body_composition_measurement)",
1379                                     "oic.r.body.fat.bodyfat =
1380 int.from_bytes(body_composition_measurement[ length - 4 : length - 2], 'big')",
1381                                     "oic.r.body.fat.units = '%'"
1382                                 ],
1383                                 "x-from-ocf": [
1384                                     "N/A"
1385                                 ]
1386                             }
1387                         }
1388                     },
1389                     "body_composition_measurement[ length - bwm_len - slm_len - ffm_len
1390 - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len - ffm_len -
1391 mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 ]" : {
1392                         "$ref": "#/definitions/byteArray",
1393                         "description": "Body Water Mass",
1394                         "x-ocf-conversion": {
1395                             "x-ocf-alias": "oic.r.body.water",
1396                             "x-to-ocf": [
1397                                 "length = len(body_composition_measurement)",
1398                                 "flags_upperbyte =
1399 body_composition_measurement[length - 2]",
1400                                 "flags_lowerbyte =
1401 body_composition_measurement[length - 1]"

```

```

1402                                     "timestamp_len = 7 if (flags_lowerbyte &
1403 0x02) else 0",
1404                                     "userid_len = 1 if (flags_lowerbyte & 0x04)
1405 else 0",
1406                                     "basal_len = 2 if (flags_lowerbyte & 0x08)
1407 else 0",
1408                                     "muscle_len = 2 if (flags_lowerbyte & 0x10)
1409 else 0",
1410                                     "mm_len = 2 if (flags_lowerbyte & 0x20) else
1411 0",
1412                                     "ffm_len = 2 if (flags_lowerbyte & 0x40) else
1413 0",
1414                                     "slm_len = 2 if (flags_lowerbyte & 0x80) else
1415 0",
1416                                     "bwm_len = 2 if (flags_upperbyte & 0x01) else
1417 0",
1418                                     "if (flags_lowerbyte & 0x01): ",
1419                                     "    oic.r.body.water.bwater =
1420 int.from_bytes(body_composition_measurement[ length - bwm_len - slm_len - ffm_len - mm_len -
1421 muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len - ffm_len - mm_len -
1422 muscle_len - basal_len - userid_len - timestamp_len - 4 ], 'big')",
1423                                     "    oic.r.body.water.units = 'lb' if
1424 (flags_lowerbyte & 0x01) 'kg'"
1425                                     ],
1426                                     "x-from-ocf": [
1427                                         "N/A"
1428                                     ]
1429                                 },
1430                                 },
1431                                 "body_composition_measurement[ length - slm_len - ffm_len - mm_len -
1432 muscle_len - basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len - muscle_len -
1433 basal_len - userid_len - timestamp_len - 4 ]" : {
1434                                     "$ref": "#/definitions/byteArray",
1435                                     "description": "Soft Lean Mass",
1436                                     "x-ocf-conversion": {
1437                                         "x-ocf-alias": "oic.r.body.slm",
1438                                         "x-to-ocf": [
1439                                             "length = len(body_composition_measurement)",
1440                                             "flags_upperbyte =
1441 body_composition_measurement[length - 2]",
1442                                             "flags_lowerbyte =
1443 body_composition_measurement[length - 1]",
1444                                             "timestamp_len = 7 if (flags_lowerbyte &
1445 0x02) else 0",
1446                                             "userid_len = 1 if (flags_lowerbyte & 0x04)
1447 else 0",
1448                                             "basal_len = 2 if (flags_lowerbyte & 0x08)
1449 else 0",
1450                                             "muscle_len = 2 if (flags_lowerbyte & 0x10)
1451 else 0",
1452                                             "mm_len = 2 if (flags_lowerbyte & 0x20) else
1453 0",
1454                                             "ffm_len = 2 if (flags_lowerbyte & 0x40) else
1455 0",
1456                                             "slm_len = 2 if (flags_lowerbyte & 0x80) else
1457 0",
1458                                             "if (flags_lowerbyte & 0x01): ",
1459                                             "    oic.r.body.slm.bwater =
1460 int.from_bytes(body_composition_measurement[ length - slm_len - ffm_len - mm_len - muscle_len -
1461 basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len - muscle_len - basal_len -
1462 userid_len - timestamp_len - 4 ], 'big')",
1463                                             "    oic.r.body.slm.units = 'lb' if
1464 (flags_lowerbyte & 0x01) 'kg'"
1465                                             ],
1466                                             "x-from-ocf": [
1467                                                 "N/A"
1468                                             ]
1469                                         }
1470                                 },
1471                                 "body_composition_measurement[ length - ffm_len - mm_len -
1472 muscle_len - basal_len - userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len

```

```

1473 - userid_len - timestamp_len - 4 ]" : {
1474     "$ref": "#/definitions/byteArray",
1475     "description": "Fat Free Mass",
1476     "x-ocf-conversion": {
1477         "x-ocf-alias": "oic.r.body.ffm",
1478         "x-to-ocf": [
1479             "length = len(body_composition_measurement)",
1480             "flags_upperbyte =
1481 body_composition_measurement[length - 2]",
1482             "flags_lowerbyte =
1483 body_composition_measurement[length - 1]",
1484             "timestamp_len = 7 if (flags_lowerbyte &
1485 0x02) else 0",
1486             "userid_len = 1 if (flags_lowerbyte & 0x04)
1487 else 0",
1488             "basal_len = 2 if (flags_lowerbyte & 0x08)
1489 else 0",
1490             "muscle_len = 2 if (flags_lowerbyte & 0x10)
1491 else 0",
1492             "mm_len = 2 if (flags_lowerbyte & 0x20) else
1493 0",
1494             "ffm_len = 2 if (flags_lowerbyte & 0x40) else
1495 0",
1496             "if (flags_lowerbyte & 0x01): ",
1497             "    oic.r.body.ffm.bwater =
1498 int.from_bytes(body_composition_measurement[ length - ffm_len - mm_len - muscle_len - basal_len -
1499 userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len - userid_len -
1500 timestamp_len - 4 ], 'big')",
1501             "    oic.r.body.ffm.units = 'lb' if
1502 (flags_lowerbyte & 0x01) 'kg'"
1503         ],
1504         "x-from-ocf": [
1505             "N/A"
1506         ]
1507     }
1508 }
1509 }
1510 }
1511 },
1512
1513 "type": "object",
1514
1515 "allOf": [
1516     { "$ref": "#/definitions/byte" },
1517     { "$ref": "#/definitions/byteArray" },
1518     { "$ref": "#/definitions/org.bluetooth.characteristic.weight_measurement" },
1519     { "$ref": "#/definitions/org.bluetooth.characteristic.body_composition_measurement" }
1520 ],
1521
1522 "required": [
1523     "weight_measurement[length - 3 : length - 1]"
1524 ]
1525 }
1526

```


1527
1528
1529

Annex A (Informative) BLE GATT based Data Model

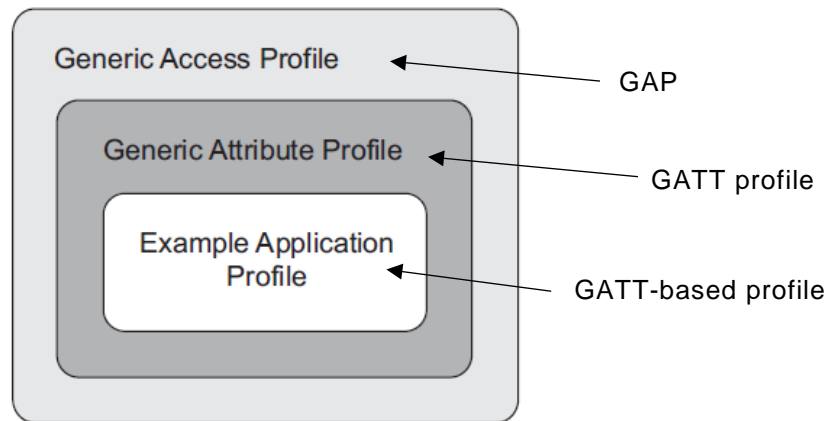
1530 A.1 BLE GATT based data model & GATT features

1531 A.1.1 Introduction

1532 The Generic Attribute Profile (GATT) defines a service framework using the Attribute Protocol.
1533 This framework defines procedures and formats of services and their characteristics. The
1534 procedures defined include discovering, reading, writing, notifying and indicating characteristics,
1535 as well as configuring the broadcast of characteristics.

1536 A.1.2 Profile dependency

1537 Figure A-1 depicts the structure and the dependencies of the profiles. A profile is dependent upon
1538 another profile if it re-uses parts of that profile by implicitly or explicitly referencing it.



1539
1540

Figure A-1 – profile dependencies

1541 A.1.3 Configurations and roles

1542 There are two roles defined in GATT profile:

- 1543 • Client: This is the device that initiates commands and requests towards the server and can
1544 receive responses, indications and notifications sent by the server.
- 1545 • Server: This is the device that accepts incoming commands and requests from the client
1546 and sends responses, indications and notifications to a client.

1547 A device can act in both roles at the same time.

1548 A.1.4 GATT profile hierarchy

1549 A.1.4.1 Introduction

1550 The GATT Profile specifies the structure in which profile data is exchanged. This structure defines
1551 basic elements such as services and characteristics, used in a profile. All of the elements are
1552 contained by Attributes. Attributes used in the ATT are containers that carry this profile data.

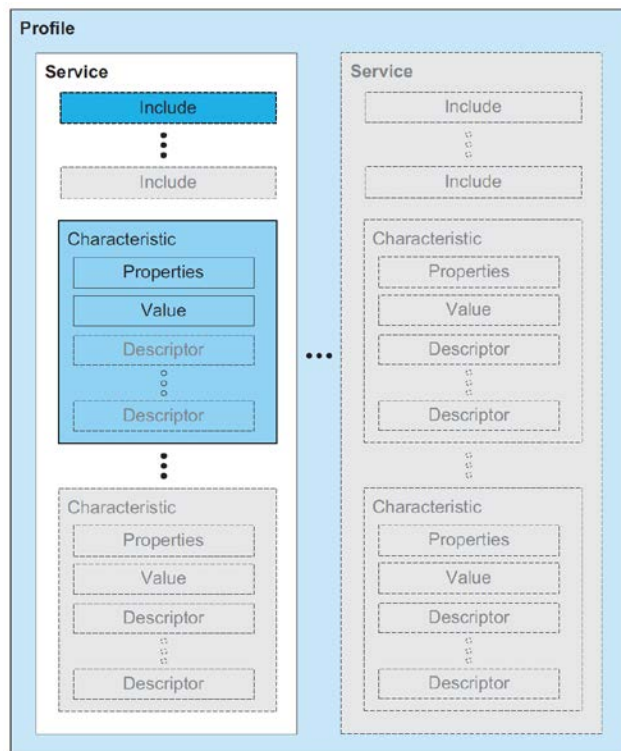
1553 The top level of the hierarchy is a profile. A profile is composed of one or more services necessary
1554 to fulfil a use case. A service is composed of characteristics or references to other services. Each
1555 characteristic contains a value and may contain optional information about the value. The service

1556 and characteristic and the components of the characteristic (i.e. value and descriptors) contain the
 1557 profile data and are all stored in Attributes on the server.

1558 Under GATT profile, entity that provides Service-Characteristics data model plays “server” role
 1559 and entity that gets data from GATT server plays “client” role.

1560 There are other application profiles based on GATT profile. They are called “GATT-based profiles”.

1561 Figure A-2 Illustrates the GATT profile hierarchy.



1562

1563

Figure A-2 – GATT profile hierarchy

1564 **A.1.4.2 Characteristic**

1565 A characteristic is a value used in a service along with properties and configuration information
 1566 about how the value is accessed and information about how the value is displayed or represented.
 1567 In GATT, a characteristic is defined by its characteristic definition. A characteristic definition
 1568 contains a characteristic declaration, characteristic properties, and a value and may contain
 1569 descriptors that describe the value or permit configuration of the server with respect to the
 1570 characteristic.

1571 **A.1.4.3 GATT features**

1572 GATT profile also supports GATT features. GATT feature defines how GATT-based data
 1573 exchanges take place. Each feature is mapped to one or more sub-procedures. These sub-
 1574 procedures describe how the ATT is used to accomplish the corresponding feature, please see
 1575 Table A-1.

1576

Table A-1 – GATT Features and ATT protocol

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

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

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

1577

1578
1579
1580

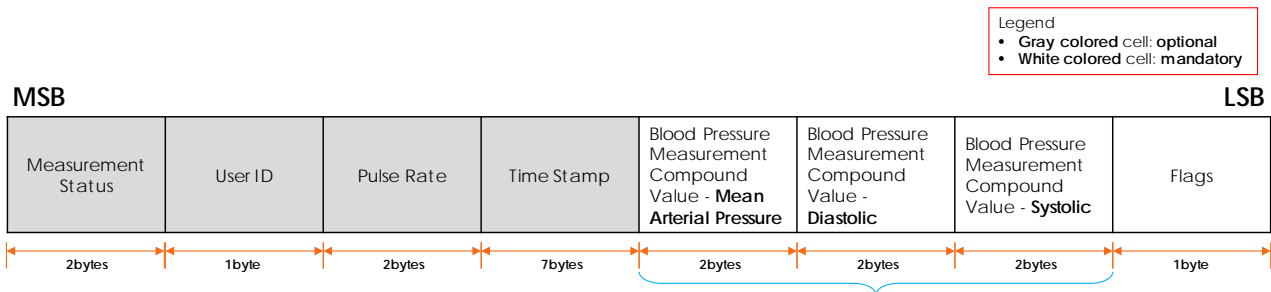
Annex B (Informative) Supporting Atomic Measurement Operation in BLE

1581 B.1 Atomic Measurement Resource Type in OCF

1582 Most OCF healthcare devices adopt the Atomic Measurement feature. Atomic Measurement
1583 Resource Type is a specialisation of a Collection to ensure that the Client can only access the
1584 Properties of the linked Resources as a single group. Thus, if an OCF device corresponding to a
1585 BLE device implements Atomic Measurement Resource Type, the BLE Bridging Function should
1586 guarantee that BLE GATT Characteristic values corresponding to properties of the Atomic
1587 Measurement Resource Type can be retrieved in atomic way.

1588 B.2 Case 1. One Characteristic covers all properties of an Atomic Measurement 1589 Resource Type

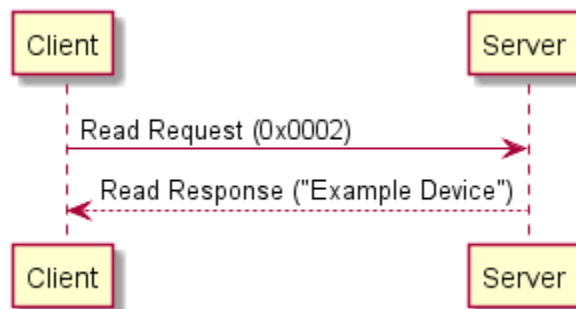
1590 In OCF-BLE mapping, a Service can be mapped to multiple OCF Resources and “a Characteristic”
1591 in a Service can be mapped to Properties in multiple OCF Resources. In general, “Value of a
1592 Characteristic” is a byte stream (see Figure B-1, byte stream is a value of “blood pressure
1593 measurement Characteristic”). Usually “value of a Characteristic” includes multiple fields like below
1594 example and each field can be mapped to a property of OCF Resource.



1595

1596 **Figure B-1 – Value of blood pressure measurement Characteristic**

1597 For blood pressure device, “blood pressure measurement Characteristic” can cover all properties
1598 in bloodpressuremonitor-am. So if BLE GATT client (OCF-BLE Bridge Platform) uses “Read
1599 Characteristic Value” operation, it can get all values corresponding to all properties in
1600 bloodpressuremonitor-am at one time (atomic operation); see Figure B-2 for an example flow.



1601

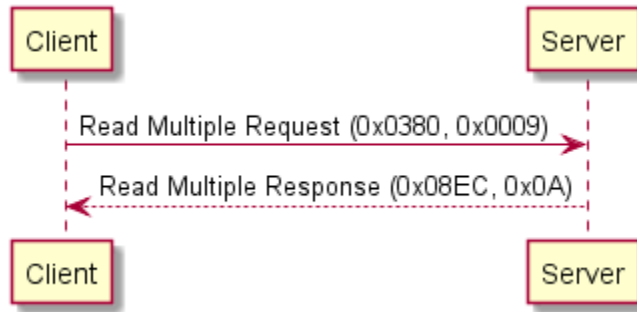
1602

1603

Figure B-2 – Read characteristic value example

1604 **B.3 Case 2. Multiple Characteristics cover all properties of an Atomic Measurement**
 1605 **Resource Type**

1606 For glucose meter, 2 Characteristics (glucose measurement Characteristic, glucose measurement
 1607 context Characteristic) cover all properties in glucosemeter-am. In this case, a BLE GATT client
 1608 (OCF-BLE Bridge Platform) can use “Read Multiple Characteristic Values” operation to get multiple
 1609 Characteristic values at one time; please see Figure B-3 for an example flow.



1610
 1611 **Figure B-3 – Read multiple characteristics value example**

1612 However, some BLE GATT server may not support all operations except for “Notification”. In this
 1613 case, a Characteristic value includes “sequence number” field, so BLE GATT client (OCF-BLE
 1614 Bridge Platform) can make a set of values which are measured at the same time by using it.

1615 Figure B-4 and Figure B-5 are two Characteristics of glucose Service.

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

1616
 1617 **Figure B-4 – Value of glucose measurement Characteristic**

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

1618
 1619 **Figure B-5 – Value of glucose measurement context Characteristic**