

OCF 2.3 – BLE Mapping Spec – BTG

Legal Disclaimer

THIS IS A DRAFT SPECIFICATION DOCUMENT ONLY AND HAS NOT BEEN ADOPTED BY THE OPEN CONNECTIVITY FOUNDATION. THIS DRAFT DOCUMENT MAY NOT BE RELIED UPON FOR ANY PURPOSE OTHER THAN REVIEW OF THE CURRENT STATE OF THE DEVELOPMENT OF THIS DRAFT DOCUMENT. THE OPEN CONNECTIVITY FOUNDATION AND ITS MEMBERS RESERVE THE RIGHT WITHOUT NOTICE TO YOU TO CHANGE ANY OR ALL PORTIONS HEREOF, DELETE PORTIONS HEREOF, MAKE ADDITIONS HERETO, DISCARD THIS DRAFT DOCUMENT IN ITS ENTIRETY OR OTHERWISE MODIFY THIS DRAFT DOCUMENT AT ANY TIME. YOU SHOULD NOT AND MAY NOT RELY UPON THIS DRAFT DOCUMENT IN ANY WAY, INCLUDING BUT NOT LIMITED TO THE DEVELOPMENT OF ANY PRODUCTS OR SERVICES. IMPLEMENTATION OF THIS DRAFT DOCUMENT IS DONE AT YOUR OWN RISK AMEND AND IT IS NOT SUBJECT TO ANY LICENSING GRANTS OR COMMITMENTS UNDER THE OPEN CONNECTIVITY FOUNDATION INTELLECTUAL PROPERTY RIGHTS POLICY OR OTHERWISE. IN CONSIDERATION OF THE OPEN CONNECTIVITY FOUNDATION GRANTING YOU ACCESS TO THIS DRAFT DOCUMENT, YOU DO HEREBY WAIVE ANY AND ALL CLAIMS ASSOCIATED HERewith INCLUDING BUT NOT LIMITED TO THOSE CLAIMS DISCUSSED BELOW, AS WELL AS CLAIMS OF DETRIMENTAL RELIANCE.

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 © 2018 Open Connectivity Foundation, Inc. All rights reserved.

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



25

26

CONTENTS

27	1	Scope.....	6
28	2	Normative references	6
29	3	Terms, definitions and symbols	6
30	3.1	Terms and definitions.....	7
31	3.2	Symbols and abbreviations	7
32	4	Document conventions and organization.....	7
33	4.1	Introduction.....	7
34	4.2	Conventions.....	7
35	4.3	Notation	7
36	4.4	Data types	8
37	5	Theory of Operation.....	8
38	5.1	Interworking Approach	8
39	5.2	Mapping Syntax	8
40	6	Device Type Mapping	9
41	6.1	Introduction.....	9
42	6.2	BLE Profile to OCF Device Types	9
43	7	BLE Profile to Resource Equivalence	10
44	7.1	Introduction.....	10
45	7.2	BLE Services to OCF Resources	10
46	8	Detailed Mappings.....	11
47	8.1	Blood Pressure Monitor Mapping	11
48	8.1.1	Introduction	11
49	8.1.2	Derived Model	11
50	8.1.3	Property Definition.....	11
51	8.1.4	Derived Model Definition.....	11
52	8.2	Glucose Meter Mapping	15
53	8.2.1	Introduction	15
54	8.2.2	Derived Model	15
55	8.2.3	Property Definition.....	15
56	8.2.4	Derived Model Definition.....	18
57	8.3	Body Thermometer Mapping	24
58	8.3.1	Introduction	24
59	8.3.2	Derived Model	24
60	8.3.3	Property Definition.....	24
61	8.3.4	Derived Model Definition.....	25
62	8.4	Body Scale Mapping	26
63	8.4.1	Introduction	26
64	8.4.2	Derived Model	27
65	8.4.3	Property Definition.....	27
66	8.4.4	Derived Model Definition.....	29



67
68



69

Figures

70

71

No table of figures entries found.



72

Tables

73	Table 1 Translation rule between BLE and OCF data model	8
74	Table 2 BLE Profile to OCF Device Type Mapping	9
75	Table 3 BLE Services to OCF Resource Type Mapping.....	10
76	Table 4 The property mapping for	
77	org.bluetooth.characteristic.blood_pressure_measurement.....	11
78	Table 5 The property mapping for	
79	org.bluetooth.characteristic.glucose_measurement_context.....	15
80	Table 6 The property mapping for org.bluetooth.characteristic.glucose_measurement	17
81	Table 7 The property mapping for org.bluetooth.characteristic.temperature_measurement ...	24
82	Table 8 The property mapping for org.bluetooth.characteristic.weight_measurement	27
83	Table 9 The property mapping for	
84	org.bluetooth.characteristic.body_composition_measurement	27
85		
86		



87 **1 Scope**

88 BLE to OCF Resources Mapping specification provides detailed mapping information between BLE
89 Profile and OCF Resources.

90 BLE Bridge is Asymmetric Server Bridge, therefore this specification provides unidirectional
91 mapping for Device Types (BLE Profile to OCF Device), identifies equivalent OCF Resources for
92 specific BLE Services, and defines the detailed Property by Property mapping using OCF defined
93 extensions to JSON schema to programmatically define the mappings.

94 **2 Normative references**

95 The following documents, in whole or in part, are normatively referenced in this document and are
96 indispensable for its application. For dated references, only the edition cited applies. For undated
97 references, the latest edition of the referenced document (including any amendments) applies.

98 JSON Schema Core, *JSON Schema: core definitions and terminology*, January 2013
99 <http://json-schema.org/latest/json-schema-core.html>

100 JSON Schema Validation, *JSON Schema: interactive and non-interactive validation*, January 2013
101 <http://json-schema.org/latest/json-schema-validation.html>

102 JSON Hyper-Schema, *JSON Hyper-Schema: A Vocabulary for Hypermedia Annotation of JSON*,
103 October 2016
104 <http://json-schema.org/latest/json-schema-hypermedia.html>

105 OCF Core Specification, *Open Connectivity Foundation Core Specification, Version 2.0*
106 https://openconnectivity.org/specs/OCF_Core_Specification_v2.0.0.pdf

107 OCF Device Specification, *Open Connectivity Foundation Device Specification, Version 2.0*
108 https://openconnectivity.org/specs/OCF_Device_Specification_v2.0.0.pdf

109 OCF Resource Type Specification, *Open Connectivity Foundation Security Specification, Version*
110 *2.0*
111 https://openconnectivity.org/specs/OCF_Resource_Type_Specification_v1.3.0.pdf

112 OCF Security Specification, *Open Connectivity Foundation Security Specification, Version 2.0*
113 https://openconnectivity.org/specs/OCF_Security_Specification_v2.0.0.pdf

114 OCF Bridging Specification, *Open Connectivity Foundation Bridging Specification, Version 2.0*
115 https://openconnectivity.org/specs/OCF_Security_Specification_v2.0.0.pdf

116 RAML Specification, *RESTful API Modeling Language, Version 0.8*
117 <https://github.com/raml-org/raml-spec/blob/master/versions/raml-08/raml-08.md>

118 OpenAPI Specification, Version 2.0
119 <https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md>

120 Derived Models for Interoperability between IoT Ecosystems, Stevens & Merriam, March 2016
121 [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)
122 [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)

123 **3 Terms, definitions and symbols**

124 All terms and definitions as defined in the OCF Core Specification, OCF Security Specification, and
125 OCF Bridging Specification also apply to this specification.



126 **3.1 Terms and definitions**

127 As defined in the OCF Core Specification and OCF Bridging Specification with the following
128 additions.

129 **3.2 Symbols and abbreviations**

130 None defined.

131 **4 Document conventions and organization**

132 **4.1 Introduction**

133 For the purposes of this document, the terms and definitions given in OCF Core Specification and
134 OCF Security Specification apply.

135 **4.2 Conventions**

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

140 **4.3 Notation**

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

143 Required (or shall or mandatory).

- 144 – These basic features shall be implemented to comply with OIC Core Architecture. The phrases
145 “shall not”, and “PROHIBITED” indicate behaviour that is prohibited, i.e. that if performed means
146 the implementation is not in compliance.

147 Recommended (or should).

- 148 – These features add functionality supported by OIC Core Architecture and should be
149 implemented. Recommended features take advantage of the capabilities OIC Core Architecture,
150 usually without imposing major increase of complexity. Notice that for compliance testing, if a
151 recommended feature is implemented, it shall meet the specified requirements to be in
152 compliance with these guidelines. Some recommended features could become requirements in
153 the future. The phrase “should not” indicates behaviour that is permitted but not recommended.

154 Allowed (or allowed).

- 155 – These features are neither required nor recommended by OIC Core Architecture, but if the
156 feature is implemented, it shall meet the specified requirements to be in compliance with these
157 guidelines.

- 158 – Conditionally allowed (CA)The definition or behaviour depends on a condition. If the specified
159 condition is met, then the definition or behaviour is allowed, otherwise it is not allowed.

160 Conditionally required (CR)

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

164 DEPRECATED

- 165 – Although these features are still described in this specification, they should not be implemented
166 except for backward compatibility. The occurrence of a deprecated feature during operation of
167 an implementation compliant with the current specification has no effect on the



168 implementation's operation and does not produce any error conditions. Backward compatibility
 169 may require that a feature is implemented and functions as specified but it shall never be used
 170 by implementations compliant with this specification.

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

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

173 4.4 Data types

174 Data types are defined in the OCF Core Specification.

175 5 Theory of Operation

176 5.1 Interworking Approach

177 The basic scheme of interworking between BLE Profile and OCF defined Resource Types is based
 178 on the derived model syntax described in Derived Models for Interoperability. Determination of the
 179 minimum set of BLE Profiles for which equivalency is required within the OCF data model was done
 180 by listing the set of BLE Profiles required for each of the device types defined by OCF Healthcare
 181 Project.

182 Basic translation rule between BLE Service/Characteristic model and OCF Resource model is
 183 described in Table 1. One or more BLE GATT-based profiles should be mapped to one Virtual OCF
 184 Server (e.g. Health Thermometer profile (HTP) is mapped to Body Thermometer device
 185 (oic.d.bodythermometer)). A BLE Service should be mapped to one or more OCF Resources (e.g.
 186 Health Thermometer Service is mapped to Temperature (oic.r.body_temperature) and Body
 187 Location for temperature (oic.r.body.location.temperature)). Each Characteristic of BLE Service
 188 should be mapped to one or more Properties of OCF Resource (if there is no BLE Characteristic
 189 corresponding to an OCF Property, default value should be used).

190 **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

191

192 5.2 Mapping Syntax

193 Within the defined syntax for derived modelling used by this Specification there are two blocks that
 194 define the actual Property-Property equivalence or mapping. These blocks are identified by the
 195 keywords 'x-to-ocf' and 'x-from-ocf'. Derived Models for Interoperability does not define a rigid
 196 syntax for these blocks; they are free form string arrays that contain pseudo-coded mapping logic.
 197 In BLE Bridging, Python (version >= 3.0) syntax is used to describe translation rules.

198 Below json skeleton shows typical translation block used in BLE derived model.

```
199 "<BLE Service Name>" : {
200   "type": "object",
201   "properties": {
202     "<a value field in BLE Characteristic value>" : {
203       "x-ocf-conversion" : {
```




```

204     "x-ocf-alias": "<corresponding OCF Resource type>",
205     "x-to-ocf": [
206         ...
207         ...
208     ],
209     "x-from-ocf": [
210         "N/A"
211     ]
212 }
213 }
214 }
215 }

```

- 216 • <BLE Service Name>: this is fully qualified name of a BLE Service (e.g.
217 org.bluetooth.characteristic.blood_pressure_measurement)
- 218 • <a value field in BLE Characteristic value>: a Characteristic value is byte stream which is
219 composed of multiple value fields. "A value field in BLE Characteristic value" is a description
220 for one of them.
- 221 • <corresponding OCF Resource type>: an OCF Resource type which is corresponding to
222 this BLE Service.
- 223 • "N/A": in BLE Bridging, most of the BLE devices are read only. So there is no specific value
224 to be written to the BLE devices from OCF Devices. Therefore nothing is described in "x-
225 from-ocf" translation section. "N/A" is used to describe this case.

226 6 Device Type Mapping

227 6.1 Introduction

228 This Section contains the mappings to Device Types.

229 6.2 BLE Profile to OCF Device Types

230 The following table captures the equivalency mapping between BLE Profile and OCF defined
231 Device Types. The minimum Resource sets for each OCF Device is provided in Annex C Healthcare
232 Device Types of OCF Device Specification.

233 **Table 2 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

234



235 **7 BLE Profile to Resource Equivalence**

236 **7.1 Introduction**

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

239 **7.2 BLE Services to OCF Resources**

240 The following tables capture the equivalency mapping between BLE Services and OCF defined
 241 Resource Types (see OCF Resource Type Specification). Detailed Property by Property mappings
 242 are provided in Section 8.

243 **Table 3 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.ffmpeg
		oic.wk.d



Device Information Service	oic.wk.p
----------------------------	----------

244

245 **8 Detailed Mappings**

246 **8.1 Blood Pressure Monitor Mapping**

247 **8.1.1 Introduction**

248 This API defines the mapping between <BLE Blood Pressure Measurement Service> and <OCF Blood Pressure, Pulse Rate Resource>.

249

250 **8.1.2 Derived Model**

251 org.bluetooth.characteristic.blood_pressure_measurement

252 **8.1.3 Property Definition**

253 **Table 4 The property mapping for**
 254 **org.bluetooth.characteristic.blood_pressure_measurement.**

BLE Property name	OCF Resource	To OCF	From OCF
blood_pressure_measurement[length - 5 : length - 3]	oic.r.blood.pressure	length = len(blood_pressure_measurement) oic.r.blood.pressure.diastolic = ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 5 : length - 3])	N/A
blood_pressure_measurement[length - 3 : length - 1]	oic.r.blood.pressure	length = len(blood_pressure_measurement) flags = blood_pressure_measurement[length - 1] oic.r.blood.pressure.systolic = ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 3 : length - 1]) oic.r.blood.pressure.units = "mmHg" if (flags & 0x01) else "kPa"	N/A
blood_pressure_measurement[length - 7 - timestamp_len - 2 : length - 7 - timestamp_len]	oic.r.pulserate	length = len(blood_pressure_measurement) flags = blood_pressure_measurement[length - 1] timestamp_len = 7 if (flags & 0x02) else 0 oic.r.pulserate.pulserate = ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 7 - timestamp_len - 2 : length - 7 - timestamp_len])	N/A
blood_pressure_measurement[length - 7 : length - 5]	oic.r.blood.pressure	length = len(blood_pressure_measurement) oic.r.blood.pressure.map = ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 7 : length - 5])	N/A

255 **8.1.4 Derived Model Definition**

```

256 {
257   "id":
258   "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.BLP.json#",
259   "$schema": "http://json-schema.org/draft-04/schema#",
260   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
261 reserved.",
262   "title": "Blood Pressure",
263   "definitions": {
264     "byte": {
265       "type": "integer",

```



```
266         "minimum": 0,
267         "maximum": 255
268     },
269     "byteArray": {
270         "type": "array",
271         "items": { "$ref": "#/definitions/byte" },
272         "minItems": 1,
273         "uniqueItems": false
274     },
275     "org.bluetooth.characteristic.blood_pressure_measurement" : {
276         "type" : "object",
277         "properties": {
278             "blood_pressure_measurement[length - 3 : length - 1]": {
279                 "$ref": "#/definitions/byteArray",
280                 "description": "Blood Pressure Measurement Compound Value - Systolic",
281                 "x-ocf-conversion": {
282                     "x-ocf-alias": "oic.r.blood.pressure",
283                     "x-to-ocf": [
284                         "def ieee11073_Sfloat_2_Float(sfloat_value):",
285                         "# reserved value for Infinity or NaN (Not a Number)",
286                         "reserved_float_values = {",
287                         "0x07FE:math.inf, # +INFINITY",
288                         "0x07FF:math.nan, # NaN (Not a Number)",
289                         "0x0800:math.nan, # NRes (Not at this Resolution)",
290                         "0x0801:math.nan, # Reserved for future",
291                         "0x0802:-math.inf # -INFINITY",
292                         "}",
293                         "mantissa = sfloat_value & 0x0FFF",
294                         "exponent = sfloat_value >> 12",
295                         "if (exponent >= 0x0008):",
296                             "exponent = -((0x000F + 1) - exponent)",
297                         "output = 0",
298                         "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
299                             "output = reserved_float_values[mantissa]",
300                         "else:",
301                             "if (mantissa >= 0x0800):",
302                                 "mantissa = -((0x0FFF + 1) - mantissa)",
303                                 "magnitude = pow(10.0, exponent)",
304                                 "output = (mantissa * magnitude)",
305                             "return output",
306                         "length = len(blood_pressure_measurement)",
307                         "flags = blood_pressure_measurement[length - 1]",
308                         "oic.r.blood.pressure.systolic =
309 ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 3 : length - 1])",
310                         "oic.r.blood.pressure.units = \"mmHg\" if (flags & 0x01) else
311 \"kPa\"",
312                     ],
313                     "x-from-ocf": [
314                         "N/A"
315                     ]
316                 }
317             },
318             "blood_pressure_measurement[length - 5 : length - 3]": {
319                 "$ref": "#/definitions/byteArray",
320                 "description": "Blood Pressure Measurement Compound Value - Diastolic",
321                 "x-ocf-conversion": {
322                     "x-ocf-alias": "oic.r.blood.pressure",
323                     "x-to-ocf": [
324                         "def ieee11073_Sfloat_2_Float(sfloat_value):",
325                         "# reserved value for Infinity or NaN (Not a Number)",
326                         "reserved_float_values = {",
327                         "0x07FE:math.inf, # +INFINITY",
328                         "0x07FF:math.nan, # NaN (Not a Number)",
```



```
329         "0x0800:math.nan, # NRes (Not at this Resolution)",
330         "0x0801:math.nan, # Reserved for future",
331         "0x0802:-math.inf # -INFINITY",
332     "}",
333     "mantissa = sfloat_value & 0x0FFF",
334     "exponent = sfloat_value >> 12",
335     "if (exponent >= 0x0008):",
336         "exponent = -((0x000F + 1) - exponent)",
337     "output = 0",
338     "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
339         "output = reserved_float_values[mantissa]",
340     "else:",
341         "if (mantissa >= 0x0800):",
342             "mantissa = -((0x0FFF + 1) - mantissa)",
343             "magnitude = pow(10.0, exponent)",
344             "output = (mantissa * magnitude)",
345         "return output",
346     "length = len(blood_pressure_measurement)",
347     "oic.r.blood.pressure.diastolic =
348 ieee11073_sfloat_2_Float(blood_pressure_measurement[length - 5 : length - 3])"
349     ],
350     "x-from-ocf": [
351         "N/A"
352     ]
353     },
354 },
355 "blood_pressure_measurement[length - 7 : length - 5]": {
356     "$ref": "#/definitions/byteArray",
357     "description": "Blood Pressure Measurement Compound Value - Mean Arterial
358 Pressure",
359     "x-ocf-conversion": {
360         "x-ocf-alias": "oic.r.blood.pressure",
361         "x-to-ocf": [
362             "def ieee11073_sfloat_2_Float(sfloat_value):",
363                 "# reserved value for Infinity or NaN (Not a Number)",
364                 "reserved_float_values = {",
365                     "0x07FE:math.inf, # +INFINITY",
366                     "0x07FF:math.nan, # NaN (Not a Number)",
367                     "0x0800:math.nan, # NRes (Not at this Resolution)",
368                     "0x0801:math.nan, # Reserved for future",
369                     "0x0802:-math.inf # -INFINITY",
370                 "}",
371                 "mantissa = sfloat_value & 0x0FFF",
372                 "exponent = sfloat_value >> 12",
373                 "if (exponent >= 0x0008):",
374                     "exponent = -((0x000F + 1) - exponent)",
375                 "output = 0",
376                 "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
377                     "output = reserved_float_values[mantissa]",
378                 "else:",
379                     "if (mantissa >= 0x0800):",
380                         "mantissa = -((0x0FFF + 1) - mantissa)",
381                         "magnitude = pow(10.0, exponent)",
382                         "output = (mantissa * magnitude)",
383                     "return output",
384                 "length = len(blood_pressure_measurement)",
385                 "oic.r.blood.pressure.map =
386 ieee11073_sfloat_2_Float(blood_pressure_measurement[length - 7 : length - 5])"
387             ],
388             "x-from-ocf": [
389                 "N/A"
390             ]
391         }
392     }
393 }
```



```
392     },
393     "blood_pressure_measurement[length - 7 - timestamp_len - 2 : length - 7 -
394 timestamp_len]": {
395         "$ref": "#/definitions/byteArray",
396         "description": "Pulse Rate",
397         "x-ocf-conversion": {
398             "x-ocf-alias": "oic.r.pulserate",
399             "x-to-ocf": [
400                 "def ieee11073_Sfloat_2_Float(sfloat_value):",
401                 "# reserved value for Infinity or NaN (Not a Number)",
402                 "reserved_float_values = {",
403                 "    0x07FE:math.inf, # +INFINITY",
404                 "    0x07FF:math.nan, # NaN (Not a Number)",
405                 "    0x0800:math.nan, # NRes (Not at this Resolution)",
406                 "    0x0801:math.nan, # Reserved for future",
407                 "    0x0802:-math.inf # -INFINITY",
408                 "}",
409                 "mantissa = sfloat_value & 0x0FFF",
410                 "exponent = sfloat_value >> 12",
411                 "if (exponent >= 0x0008):",
412                 "    exponent = -((0x000F + 1) - exponent)",
413                 "output = 0",
414                 "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
415                 "    output = reserved_float_values[mantissa]",
416                 "else:",
417                 "    if (mantissa >= 0x0800):",
418                 "        mantissa = -((0x0FFF + 1) - mantissa)",
419                 "        magnitude = pow(10.0, exponent)",
420                 "        output = (mantissa * magnitude)",
421                 "return output",
422                 "length = len(blood_pressure_measurement)",
423                 "flags = blood_pressure_measurement[length - 1]",
424                 "timestamp_len = 7 if (flags & 0x02) else 0",
425                 "oic.r.pulserate.pulserate =
426 ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 7 - timestamp_len - 2 :
427 length - 7 - timestamp_len])",
428             ],
429             "x-from-ocf": [
430                 "N/A"
431             ]
432         }
433     }
434 }
435 }
436 },
437
438 "type": "object",
439
440 "allof": [
441     { "$ref": "#/definitions/byte" },
442     { "$ref": "#/definitions/byteArray" },
443     { "$ref": "#/definitions/org.bluetooth.characteristic.blood_pressure_measurement" }
444 ],
445
446 "required": [
447     "blood_pressure_measurement[length - 3 : length - 1]",
448     "blood_pressure_measurement[length - 5 : length - 3]"
449 ]
450 }
```



451 **8.2 Glucose Meter Mapping**

452 **8.2.1 Introduction**

453 This API defines the mapping between <BLE Glucose Measurement Service, BLE Glucose
 454 Measurement Context Service> and <OCF Glucose Resource, OCF Glucose Sample Location
 455 Resource, OCF Context Carbohydrates Resource, OCF Context Exercise Resource, OCF Glucose
 456 Hemoglobin Bound to Glucose A1c Form (HbA1c) Resource, OCF Context Health Resource, OCF
 457 Context Tester Resource, OCF Context Meal Resource, OCF Context Medication Resource>.

458 **8.2.2 Derived Model**

459 org.bluetooth.characteristic.glucose_measurement_context,
 460 org.bluetooth.characteristic.glucose_measurement

461 **8.2.3 Property Definition**

462 **Table 5 The property mapping for**
 463 **org.bluetooth.characteristic.glucose_measurement_context.**

BLE Property name	OCF Resource	To OCF	From OCF
glucose_measurement_context[length - health_len - meal_len - carb_len - extflags_len - 3]	oic.r.glucose.tester	length = len(glucose_measurement_context) flags = glucose_measurement_context[length - 1] extflags_len = 1 if (flags & 0x80) else 0 carb_len = 3 if (flags & 0x01) else 0 meal_len = 1 if (flags & 0x02) else 0 health_len = 1 if (flags & 0x04) else 0 if (flags & 0x04): 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 - carb_len - extflags_len - 3 : length - 1 - extflags_len - 3]	oic.r.glucose.carb	length = len(glucose_measurement_context) flags = glucose_measurement_context[length - 1] extflags_len = 1 if (flags & 0x80) else 0 carb_len = 3 if (flags & 0x01) else 0 if (flags & 0x01): oic.r.glucose.carb.carb = ieee11073_Sfloat_2_Float(glucose_measurement_context[length - carb_len - extflags_len - 3 : length - 1 - extflags_len - 3]) * 1000	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 0 carb_len = 3 if (flags & 0x01) else 0 meal_len = 1 if (flags & 0x02) else 0 if (flags & 0x02): meal = { 1:'preprandial', 2:'postprandial', 3:'fasting', 4:'casual', 5:'bedtime' } oic.r.glucose.meal.meal = meal[glucose_measurement_context[length - meal_len - carb_len - extflags_len - 3]]	N/A



<p>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]</p>	<p>oic.r.glucose.hba1c</p>	<p>length = len(glucose_measurement_context) flags = glucose_measurement_context[length - 1] extflags_len = 1 if (flags & 0x80) else 0 carb_len = 3 if (flags & 0x01) else 0 meal_len = 1 if (flags & 0x02) else 0 health_len = 1 if (flags & 0x04) else 0 exercise_len = 3 if (flags & 0x08) else 0 medication_len = 3 if (flags & 0x10) else 0 hba1c_len = 2 if (flags & 0x40) else 0 if (flags & 0x40): oic.r.glucose.hba1c.hba1c = ieee11073_Sfloat_2_Float(glucose_measurement_context[length - hba1c_len - medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3])</p>	<p>N/A</p>
<p>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]</p>	<p>oic.r.glucose.medication</p>	<p>length = len(glucose_measurement_context) flags = glucose_measurement_context[length - 1] extflags_len = 1 if (flags & 0x80) else 0 carb_len = 3 if (flags & 0x01) else 0 meal_len = 1 if (flags & 0x02) else 0 health_len = 1 if (flags & 0x04) else 0 exercise_len = 3 if (flags & 0x08) else 0 medication_len = 3 if (flags & 0x10) else 0 hba1c_len = 2 if (flags & 0x40) else 0 if (flags & 0x10): medication = ieee11073_Sfloat_2_Float(glucose_measurement_context[length - medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - 1 - exercise_len - health_len - meal_len - carb_len - extflags_len - 3]) oic.r.glucose.medication.medication = medication * 1000 oic.r.glucose.medication.units = 'mL' if (flags & 0x20) else 'mg'</p>	<p>N/A</p>
<p>glucose_measurement_context[length - 2 - health_len - meal_len - carb_len - extflags_len - 3]</p>	<p>oic.r.glucose.exercise</p>	<p>length = len(glucose_measurement_context) flags = glucose_measurement_context[length - 1] extflags_len = 1 if (flags & 0x80) else 0 carb_len = 3 if (flags & 0x01) else 0 meal_len = 1 if (flags & 0x02) else 0 health_len = 1 if (flags & 0x04) else 0 if (flags & 0x08): oic.r.glucose.exercise.exercise = glucose_measurement_context[length - 2 - health_len - meal_len - carb_len - extflags_len - 3]</p>	<p>N/A</p>



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 0 carb_len = 3 if (flags & 0x01) else 0 meal_len = 1 if (flags & 0x02) else 0 health_len = 1 if (flags & 0x04) else 0 exercise_len = 3 if (flags & 0x08) else 0 if (flags & 0x10): regimen = { 1:'rapidacting', 2:'shortacting', 3:'intermediateacting', 4:'longacting', 5:'premix' } oic.r.glucose.medication.regimen = regimen[glucose_measurement_context[length - 1 - exercise_len - health_len - meal_len - carb_len - extflags_len - 3]]	N/A
glucose_measurement_context[length - 1 - extflags_len - 3]	oic.r.glucose.carb	length = len(glucose_measurement_context) flags = glucose_measurement_context[length - 1] extflags_len = 1 if (flags & 0x80) else 0 if (flags & 0x01): meal = { 1:'breakfast', 2:'lunch', 3:'dinner', 4:'snack', 5:'drink', 6:'supper', 7:'brunch' } oic.r.glucose.carb.meal = meal[glucose_measurement_context[length - 1 - extflags_len - 3]]	N/A

464

465

Table 6 The property mapping for org.bluetooth.characteristic.glucose_measurement

BLE name	Property	OCF Resource	To OCF	From OCF
glucose_measurement[length - 1 - 2 - timeoffset_len - 10]		oic.r.glucose.samplelocation	length = len(glucose_measurement) flags = glucose_measurement[length - 1] timeoffset_len = 2 if (flags & 0x01) else 0 if (flags & 0x02): samplelocation = { 1:'finger', 2:'ast', 3:'earlobe', 4:'ctrlsolution' } oic.r.glucose.samplelocation.samplelocation = samplelocation[glucose_measurement[length - 1 - 2 - timeoffset_len - 10] & 0xf0]	N/A
glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len - 10]		oic.r.glucose	length = len(glucose_measurement) flags = glucose_measurement[length - 1] timeoffset_len = 2 if (flags & 0x01) else 0 if (flags & 0x02) == True: glucose = ieee11073_Sfloat_2_Float(glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len - 10]) oic.r.glucose.glucose = (glucose * 1000) if (flags & 0x04) else (glucose * 0.1 * 1000 * 1000) oic.r.glucose.units = 'mmol/L' if (flags & 0x04) else 'mg/dL' if (flags & 0x02) == False: oic.r.glucose.glucose = 0 oic.r.glucose.units = 'mmol/L'	N/A



466 8.2.4 Derived Model Definition

```
467 {
468   "id":
469   "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.GLP.json#",
470   "$schema": "http://json-schema.org/draft-04/schema#",
471   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
472 reserved.",
473   "title": "Glucose",
474   "definitions": {
475     "byte": {
476       "type": "integer",
477       "minimum": 0,
478       "maximum": 255
479     },
480     "byteArray": {
481       "type": "array",
482       "items": { "$ref": "#/definitions/byte" },
483       "minItems": 1,
484       "uniqueItems": false
485     },
486     "org.bluetooth.characteristic.glucose_measurement": {
487       "type": "object",
488       "properties": {
489         "glucose_measurement[length - 2 - timeoffset_len - 10 : length -
490 timeoffset_len - 10]": {
491           "$ref": "#/definitions/byteArray",
492           "description": "Glucose Concentration",
493           "x-ocf-conversion": {
494             "x-ocf-alias": "oic.r.glucose",
495             "x-to-ocf": [
496               "def ieee11073_Sfloat_2_Float(sfloat_value):",
497                 "# reserved value for Infinity or NaN (Not a Number)",
498                 "reserved_float_values = {",
499                   "0x07FE:math.inf, # +INFINITY",
500                   "0x07FF:math.nan, # NaN (Not a Number)",
501                   "0x0800:math.nan, # NRes (Not at this Resolution)",
502                   "0x0801:math.nan, # Reserved for future",
503                   "0x0802:-math.inf # -INFINITY",
504                 "}",
505                 "mantissa = sfloat_value & 0x0FFF",
506                 "exponent = sfloat_value >> 12",
507                 "if (exponent >= 0x0008):",
508                   "exponent = -((0x000F + 1) - exponent)",
509                 "output = 0",
510                 "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
511                   "output = reserved_float_values[mantissa]",
512                 "else:",
513                   "if (mantissa >= 0x0800):",
514                     "mantissa = -((0x0FFF + 1) - mantissa)",
515                     "magnitude = pow(10.0, exponent)",
516                     "output = (mantissa * magnitude)",
517                   "return output",
518                 "length = len(glucose_measurement)",
519                 "flags = glucose_measurement[length - 1]",
520                 "timeoffset_len = 2 if (flags & 0x01) else 0",
521                 "if (flags & 0x02) == True:",
522                   "glucose = ieee11073_Sfloat_2_Float(glucose_measurement[length - 2
523 - timeoffset_len - 10 : length - timeoffset_len - 10])",
524                   "oic.r.glucose.glucose = (glucose * 1000) if (flags & 0x04) else
525 (glucose * 0.1 * 1000 * 1000)",
526                   "oic.r.glucose.units = 'mmol/L' if (flags & 0x04) else 'mg/dL'",
```



```
528         "if (flags & 0x02) == False:",
529         "    oic.r.glucose.glucose = 0",
530         "    oic.r.glucose.units = 'mmol/L'"
531     ],
532     "x-from-ocf": [
533         "N/A"
534     ]
535 }
536 },
537 "glucose_measurement[length - 1 - 2 - timeoffset_len - 10]": {
538     "$ref": "#/definitions/byteArray",
539     "description": "Sample Location",
540     "x-ocf-conversion": {
541         "x-ocf-alias": "oic.r.glucose.samplelocation",
542         "x-to-ocf": [
543             "length = len(glucose_measurement)",
544             "flags = glucose_measurement[length - 1]",
545             "timeoffset_len = 2 if (flags & 0x01) else 0",
546             "if (flags & 0x02):",
547                 "    samplelocation = { 1:'finger', 2:'ast', 3:'earlobe',
548 4:'ctrlsolution' }",
549             "    oic.r.glucose.samplelocation.samplelocation =
550 samplelocation[glucose_measurement[length - 1 - 2 - timeoffset_len - 10] & 0xf0]"
551         ],
552         "x-from-ocf": [
553             "N/A"
554         ]
555     }
556 }
557 },
558 },
559
560 "org.bluetooth.characteristic.glucose_measurement_context": {
561     "type": "object",
562     "properties": {
563         "glucose_measurement_context[length - carb_len - extflags_len - 3 : length -
564 1 - extflags_len - 3]": {
565             "$ref": "#/definitions/byteArray",
566             "description": "Carbohydrate",
567             "x-ocf-conversion": {
568                 "x-ocf-alias": "oic.r.glucose.carb",
569                 "x-to-ocf": [
570                     "def ieee11073_Sfloat_2_Float(sfloat_value):",
571                         "# reserved value for Infinity or NaN (Not a Number)",
572                         "reserved_float_values = {",
573                             "0x07FE:math.inf, # +INFINITY",
574                             "0x07FF:math.nan, # NaN (Not a Number)",
575                             "0x0800:math.nan, # NRes (Not at this Resolution)",
576                             "0x0801:math.nan, # Reserved for future",
577                             "0x0802:-math.inf # -INFINITY",
578                         "}",
579                         "mantissa = sfloat_value & 0x0FFF",
580                         "exponent = sfloat_value >> 12",
581                         "if (exponent >= 0x0008):",
582                             "exponent = -((0x000F + 1) - exponent)",
583                         "output = 0",
584                         "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
585                             "output = reserved_float_values[mantissa]",
586                         "else:",
587                             "if (mantissa >= 0x0800):",
588                                 "mantissa = -((0x0FFF + 1) - mantissa)",
589                                 "magnitude = pow(10.0, exponent)",
590                                 "output = (mantissa * magnitude)",
```



```
591         "return output",
592         "length = len(glucose_measurement_context)",
593         "flags = glucose_measurement_context[length - 1]",
594         "extflags_len = 1 if (flags & 0x80) else 0",
595         "carb_len = 3 if (flags & 0x01) else 0",
596         "if (flags & 0x01): ",
597         "    oic.r.glucose.carb.carb =
598 ieee11073_Sfloat_2_Float(glucose_measurement_context[length - carb_len - extflags_len -
599 3 : length - 1 - extflags_len - 3]) * 1000"
600     ],
601     "x-from-ocf": [
602         "N/A"
603     ]
604 }
605 },
606 "glucose_measurement_context[length - 1 - extflags_len - 3]": {
607     "$ref": "#/definitions/byteArray",
608     "description": "Carbohydrate ID",
609     "x-ocf-conversion": {
610         "x-ocf-alias": "oic.r.glucose.carb",
611         "x-to-ocf": [
612             "length = len(glucose_measurement_context)",
613             "flags = glucose_measurement_context[length - 1]",
614             "extflags_len = 1 if (flags & 0x80) else 0",
615             "if (flags & 0x01): ",
616             "    meal = { 1:'breakfast', 2:'lunch', 3:'dinner', 4:'snack',
617 5:'drink', 6:'supper', 7:'brunch' }",
618             "    oic.r.glucose.carb.meal = meal[glucose_measurement_context[length
619 - 1 - extflags_len - 3]]"
620         ],
621         "x-from-ocf": [
622             "N/A"
623         ]
624     }
625 },
626 "glucose_measurement_context[length - 2 - health_len - meal_len - carb_len -
627 extflags_len - 3]": {
628     "$ref": "#/definitions/byteArray",
629     "description": "Exercise Intensity",
630     "x-ocf-conversion": {
631         "x-ocf-alias": "oic.r.glucose.exercise",
632         "x-to-ocf": [
633             "length = len(glucose_measurement_context)",
634             "flags = glucose_measurement_context[length - 1]",
635             "extflags_len = 1 if (flags & 0x80) else 0",
636             "carb_len = 3 if (flags & 0x01) else 0",
637             "meal_len = 1 if (flags & 0x02) else 0",
638             "health_len = 1 if (flags & 0x04) else 0",
639             "if (flags & 0x08): ",
640             "    oic.r.glucose.exercise.exercise =
641 glucose_measurement_context[length - 2 - health_len - meal_len - carb_len - extflags_len
642 - 3]"
643         ],
644         "x-from-ocf": [
645             "N/A"
646         ]
647     }
648 },
649 "glucose_measurement_context[length - hba1c_len - medication_len -
650 exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length -
651 medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3]": {
652     "$ref": "#/definitions/byteArray",
653     "description": "HbA1c",
```



```
654 "x-ocf-conversion": {
655   "x-ocf-alias": "oic.r.glucose.hbabc",
656   "x-to-ocf": [
657     "def ieee11073_Sfloat_2_Float(sfloat_value):",
658     "# reserved value for Infinity or NaN (Not a Number)",
659     "reserved_float_values = {",
660     "  0x07FE:math.inf, # +INFINITY",
661     "  0x07FF:math.nan, # NaN (Not a Number)",
662     "  0x0800:math.nan, # NRes (Not at this Resolution)",
663     "  0x0801:math.nan, # Reserved for future",
664     "  0x0802:-math.inf # -INFINITY",
665     "}",
666     "mantissa = sfloat_value & 0x0FFF",
667     "exponent = sfloat_value >> 12",
668     "if (exponent >= 0x0008):",
669     "  exponent = -(0x000F + 1) - exponent",
670     "output = 0",
671     "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
672     "  output = reserved_float_values[mantissa]",
673     "else:",
674     "  if (mantissa >= 0x0800):",
675     "    mantissa = -(0x0FFF + 1) - mantissa",
676     "    magnitude = pow(10.0, exponent)",
677     "    output = (mantissa * magnitude)",
678     "  return output",
679     "length = len(glucose_measurement_context)",
680     "flags = glucose_measurement_context[length - 1]",
681     "extflags_len = 1 if (flags & 0x80) else 0",
682     "carb_len = 3 if (flags & 0x01) else 0",
683     "meal_len = 1 if (flags & 0x02) else 0",
684     "health_len = 1 if (flags & 0x04) else 0",
685     "exercise_len = 3 if (flags & 0x08) else 0",
686     "medication_len = 3 if (flags & 0x10) else 0",
687     "hbabc_len = 2 if (flags & 0x40) else 0",
688     "if (flags & 0x40):",
689     "  oic.r.glucose.hbabc.hbabc =
690     ieee11073_Sfloat_2_Float(glucose_measurement_context[length - hbabc_len - medication_len
691     - exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length -
692     medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3])",
693     ],
694     "x-from-ocf": [
695     "  N/A"
696     ]
697   },
698 },
699 "glucose_measurement_context[length - health_len - meal_len - carb_len -
700 extflags_len - 3]": {
701   "$ref": "#/definitions/byteArray",
702   "description": "Health",
703   "x-ocf-conversion": {
704     "x-ocf-alias": "oic.r.glucose.health",
705     "x-to-ocf": [
706       "length = len(glucose_measurement_context)",
707       "flags = glucose_measurement_context[length - 1]",
708       "extflags_len = 1 if (flags & 0x80) else 0",
709       "carb_len = 3 if (flags & 0x01) else 0",
710       "meal_len = 1 if (flags & 0x02) else 0",
711       "health_len = 1 if (flags & 0x04) else 0",
712       "if (flags & 0x04):",
713       "  health = { 1:'minor', 2:'major', 3:'menses', 4:'stress',
714       5:'none' }",
```



```
715         " oic.r.glucose.health.health =
716 health[glucose_measurement_context[length - health_len - meal_len - carb_len -
717 extflags_len - 3] & 0xf0]"
718     ],
719     "x-from-ocf": [
720         "N/A"
721     ]
722     }
723 },
724     "glucose_measurement_context[length - health_len - meal_len - carb_len -
725 extflags_len - 3]": {
726     "$ref": "#/definitions/byteArray",
727     "description": "Tester",
728     "x-ocf-conversion": {
729         "x-ocf-alias": "oic.r.glucose.tester",
730         "x-to-ocf": [
731             "length = len(glucose_measurement_context)",
732             "flags = glucose_measurement_context[length - 1]",
733             "extflags_len = 1 if (flags & 0x80) else 0",
734             "carb_len = 3 if (flags & 0x01) else 0",
735             "meal_len = 1 if (flags & 0x02) else 0",
736             "health_len = 1 if (flags & 0x04) else 0",
737             "if (flags & 0x04): ",
738             "    tester = { 1:'self', 2:'hcp', 3:'lab' }",
739             "    oic.r.glucose.tester.tester =
740 tester[glucose_measurement_context[length - health_len - meal_len - carb_len -
741 extflags_len - 3] & 0x0f]"
742         ],
743         "x-from-ocf": [
744             "N/A"
745         ]
746     }
747 },
748     "glucose_measurement_context[length - meal_len - carb_len - extflags_len -
749 3]": {
750     "$ref": "#/definitions/byteArray",
751     "description": "Meal",
752     "x-ocf-conversion": {
753         "x-ocf-alias": "oic.r.glucose.meal",
754         "x-to-ocf": [
755             "length = len(glucose_measurement_context)",
756             "flags = glucose_measurement_context[length - 1]",
757             "extflags_len = 1 if (flags & 0x80) else 0",
758             "carb_len = 3 if (flags & 0x01) else 0",
759             "meal_len = 1 if (flags & 0x02) else 0",
760             "if (flags & 0x02): ",
761             "    meal = { 1:'preprandial', 2:'postprandial', 3:'fasting',
762 4:'casual', 5:'bedtime' }",
763             "    oic.r.glucose.meal.meal = meal[glucose_measurement_context[length
764 - meal_len - carb_len - extflags_len - 3]]"
765         ],
766         "x-from-ocf": [
767             "N/A"
768         ]
769     }
770 },
771     "glucose_measurement_context[length - medication_len - exercise_len -
772 health_len - meal_len - carb_len - extflags_len - 3 : length - 1 - exercise_len -
773 health_len - meal_len - carb_len - extflags_len - 3]": {
774     "$ref": "#/definitions/byteArray",
775     "description": "Medication",
776     "x-ocf-conversion": {
777         "x-ocf-alias": "oic.r.glucose.medication",
```



```
778 "x-to-ocf": [  
779   "def ieee11073_Sfloat_2_Float(sfloat_value):",  
780     "# reserved value for Infinity or NaN (Not a Number)",  
781     "reserved_float_values = {",  
782       "0x07FE:math.inf, # +INFINITY",  
783       "0x07FF:math.nan, # NaN (Not a Number)",  
784       "0x0800:math.nan, # NRes (Not at this Resolution)",  
785       "0x0801:math.nan, # Reserved for future",  
786       "0x0802:-math.inf # -INFINITY",  
787     "}",  
788     "mantissa = sfloat_value & 0x0FFF",  
789     "exponent = sfloat_value >> 12",  
790     "if (exponent >= 0x0008):",  
791       "exponent = -((0x000F + 1) - exponent)",  
792     "output = 0",  
793     "if (mantissa >= 0x07FE and mantissa <= 0x0802):",  
794       "output = reserved_float_values[mantissa]",  
795     "else:",  
796       "if (mantissa >= 0x0800):",  
797         "mantissa = -((0x0FFF + 1) - mantissa)",  
798         "magnitude = pow(10.0, exponent)",  
799         "output = (mantissa * magnitude)",  
800       "return output",  
801     "length = len(glucose_measurement_context)",  
802     "flags = glucose_measurement_context[length - 1]",  
803     "extflags_len = 1 if (flags & 0x80) else 0",  
804     "carb_len = 3 if (flags & 0x01) else 0",  
805     "meal_len = 1 if (flags & 0x02) else 0",  
806     "health_len = 1 if (flags & 0x04) else 0",  
807     "exercise_len = 3 if (flags & 0x08) else 0",  
808     "medication_len = 3 if (flags & 0x10) else 0",  
809     "hbalc_len = 2 if (flags & 0x40) else 0",  
810     "if (flags & 0x10): ",  
811     "  medication =  
812     ieee11073_Sfloat_2_Float(glucose_measurement_context[length - medication_len -  
813     exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - 1 -  
814     exercise_len - health_len - meal_len - carb_len - extflags_len - 3])",  
815     "  oic.r.glucose.medication.medication = medication * 1000",  
816     "  oic.r.glucose.medication.units = 'mL' if (flags & 0x20) else  
817     'mg'",  
818   ],  
819   "x-from-ocf": [  
820     "N/A"  
821   ]  
822 }  
823 },  
824 "glucose_measurement_context[length - 1 - exercise_len - health_len -  
825 meal_len - carb_len - extflags_len - 3]": {  
826   "$ref": "#/definitions/byteArray",  
827   "description": "Medication ID",  
828   "x-ocf-conversion": {  
829     "x-ocf-alias": "oic.r.glucose.medication",  
830     "x-to-ocf": [  
831       "length = len(glucose_measurement_context)",  
832       "flags = glucose_measurement_context[length - 1]",  
833       "extflags_len = 1 if (flags & 0x80) else 0",  
834       "carb_len = 3 if (flags & 0x01) else 0",  
835       "meal_len = 1 if (flags & 0x02) else 0",  
836       "health_len = 1 if (flags & 0x04) else 0",  
837       "exercise_len = 3 if (flags & 0x08) else 0",  
838       "if (flags & 0x10): ",  
839       "  regimen = { 1:'rapidacting', 2:'shortacting',  
840 3:'intermediateacting', 4:'longacting', 5:'premix' }",
```



```

841         " oic.r.glucose.medication.regimen =
842 regimen[ glucose_measurement_context[length - 1 - exercise_len - health_len - meal_len -
843 carb_len - extflags_len - 3] ]"
844     ],
845     "x-from-ocf": [
846         "N/A"
847     ]
848 }
849 }
850 }
851 }
852 },
853
854 "type": "object",
855
856 "allOf": [
857     { "$ref": "#/definitions/byte" },
858     { "$ref": "#/definitions/byteArray" },
859     { "$ref":
860 "#/definitions/org.bluetooth.characteristic.org.bluetooth.characteristic.glucose_measure
861 ment" },
862     { "$ref":
863 "#/definitions/org.bluetooth.characteristic.org.bluetooth.characteristic.glucose_measure
864 ment_context" }
865 ],
866
867 "required": [
868     "glucose_measurement[length - 2 - timeoffset_len - 10 : length - timeoffset_len -
869 10]"
870 ]
871 }

```

872 8.3 Body Thermometer Mapping

873 8.3.1 Introduction

874 This API defines the mapping between <BLE Temperature Measurement Service> and <OCF
875 Temperature Resource, OCF Body Location for temperature Resource>.

876 8.3.2 Derived Model

877 org.bluetooth.characteristic.temperature_measurement.

878 8.3.3 Property Definition

879 **Table 7** The property mapping for org.bluetooth.characteristic.temperature_measurement

BLE Property name	OCF Resource	To OCF	From OCF
temperature_measurement[length - 5 : length - 1]	oic.r.temperature	length = len(temperature_measurement) flags = temperature_measurement[length - 1] oic.r.temperature.temperature = ieee11073_Float_2_Float(temperature_measurement[length - 5 : length - 1]) oic.r.temperature.units = 'F' if (flags & 0x01) else 'C'	N/A



<pre>temperature_measurement[length - temperaturetype_len - timestamp_len - 5]</pre>	<pre>oic.r.body.location.tempe rature</pre>	<pre>length = len(temperature_measurement) flags = temperature_measurement[length - 1] timestamp_len = 7 if (flags & 0x02) 0 temperaturetype_len = 1 if (flags & 0x04) 0 if (flags & 0x04): bloc = { 1:'xxx', 2:'body', 3:'ear', 4:'finger', 5:'gastro', 6:'mouth', 7:'rectum', 8:'toe', 9:'tympanum' } oic.r.body.location.temperature.bloc = bloc[temperature_measurement[length - temperaturetype_len - timestamp_len - 5]]</pre>	<p>N/A</p>
--	---	--	------------

8.3.4 Derived Model Definition

```

881 {
882   "id":
883   "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.HTP.json#",
884   "$schema": "http://json-schema.org/draft-04/schema#",
885   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
886 reserved.",
887   "title": "Health Thermometer",
888   "definitions": {
889     "byte": {
890       "type": "integer",
891       "minimum": 0,
892       "maximum": 255
893     },
894     "byteArray": {
895       "type": "array",
896       "items": { "$ref": "#/definitions/byte" },
897       "minItems": 1,
898       "uniqueItems": false
899     },
900     "org.bluetooth.characteristic.temperature_measurement" : {
901       "type": "object",
902       "properties" : {
903         "temperature_measurement[ length - 5 : length - 1 ]" : {
904           "$ref": "#/definitions/byteArray",
905           "description": "Temperature",
906           "x-ocf-conversion": {
907             "x-ocf-alias": "oic.r.temperature",
908             "x-to-ocf": [
909               "# convert IEEE11073 FLOAT to float",
910               "def ieee11073_Float_2_Float(float_value):",
911               "# reserved value for Infinity or NaN (Not a Number)",
912               "reserved_float_values = {",
913               "0x007FFFFE:math.inf, # +INFINITY",
914               "0x007FFFFFF:math.nan, # NaN (Not a Number)",
915               "0x00800000:math.nan, # NRes (Not at this Resolution)",
916               "0x00800001:math.nan, # Reserved for future",
917               "0x00800002:-math.inf # -INFINITY",
918             "}",
919             "mantissa = float_value & 0x00FFFFFF",
920             "exponent = float_value >> 24",
921             "if (exponent >= 0x00000080):",
922               "exponent = -(0x000000FF + 1) - exponent)",
923             "output = 0",
924             "if (mantissa >= 0x007FFFFE and mantissa <= 0x00800002):",
925               "output = reserved_float_values[mantissa]",
926             "else:",
927               "if (mantissa >= 0x00800000):",
928

```



```
929         "mantissa = -((0x00FFFFFF + 1) - mantissa)",
930         "magnitude = pow(10.0, exponent)",
931         "output = (mantissa * magnitude)",
932         "return output",
933         "length = len(temperature_measurement)",
934         "flags = temperature_measurement[length - 1]",
935         "oic.r.temperature.temperature =
936 ieee11073_Float_2_Float(temperature_measurement[ length - 5 : length - 1 ])",
937         "oic.r.temperature.units = 'F' if (flags & 0x01) else 'C'"
938     ],
939     "x-from-ocf": [
940         "N/A"
941     ]
942 }
943 },
944 "temperature_measurement[ length - temperaturetype_len - timestamp_len -
945 5 ]" : {
946     "$ref": "#/definitions/byteArray",
947     "description": "Temperature Type",
948     "x-ocf-conversion": {
949         "x-ocf-alias": "oic.r.body.location.temperature",
950         "x-to-ocf": [
951             "length = len(temperature_measurement)",
952             "flags = temperature_measurement[length - 1]",
953             "timestamp_len = 7 if (flags & 0x02) 0",
954             "temperaturetype_len = 1 if (flags & 0x04) 0",
955             "if (flags & 0x04):",
956             "    bloc = { 1:'xxx', 2:'body', 3:'ear', 4:'finger', 5:'gastro',
957 6:'mouth', 7:'rectum', 8:'toe', 9:'tympnum' }",
958             "    oic.r.body.location.temperature.bloc =
959 bloc[temperature_measurement[ length - temperaturetype_len - timestamp_len - 5 ] ]"
960         ],
961         "x-from-ocf": [
962             "N/A"
963         ]
964     }
965 }
966 }
967 }
968 },
969
970 "type": "object",
971
972 "allof": [
973     { "$ref": "#/definitions/byte" },
974     { "$ref": "#/definitions/byteArray" },
975     { "$ref": "#/definitions/org.bluetooth.characteristic.temperature_measurement" }
976 ],
977
978 "required": [
979     "temperature_measurement[ length - 5 : length - 1 ]"
980 ]
981 }
```

982 8.4 Body Scale Mapping

983 8.4.1 Introduction

984 This API defines the mapping between <BLE Weight Measurement Service, BLE Body Composition
985 Measurement Service> and <OCF Weight Resource, OCF Body Mass Index (BMI) Resource, OCF
986 Height Resource, OCF Body Fat Resource, OCF Body Water Resource, OCF Body Soft Lean Mass,
987 OCF Body Fat Free Mass Resource>.



988 **8.4.2 Derived Model**

989 org.bluetooth.characteristic.weight_measurement.
990 org.bluetooth.characteristic.body_composition_measurement.

991 **8.4.3 Property Definition**

992 **Table 8 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 0 oic.r.weight.weight = int.from_bytes(weight_measurement[length - 3 : length - 1], 'big') oic.r.weight.units = 'lb' if (flags & 0x01) else 'kg'	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 0 userid_len = 1 if (flags & 0x04) else 0 if (flags & 0x08): oic.r.bmi.bmi = int.from_bytes(weight_measurement[length - 2 - userid_len - timeoffset_len - 3 : length - userid_len - timeoffset_len - 3], 'big')	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 0 userid_len = 1 if (flags & 0x04) else 0 height_len = 4 if (flags & 0x08) else 0 if (flags & 0x08): oic.r.height.height = int.from_bytes(weight_measurement[length - height_len - userid_len - timeoffset_len - 3 : length - 2 - userid_len - timeoffset_len - 3], 'big') oic.r.height.units = 'in' if (flags & 0x01) else 'm'	N/A

993 **Table 9 The property mapping for**
994 **org.bluetooth.characteristic.body_composition_measurement**

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



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



		oic.r.body.fat.bodyfat = int.from_bytes(body_composition_measurement[length h - 4 : length - 2], 'big')	
		oic.r.body.fat.units = '%'	

995 **8.4.4 Derived Model Definition**

```

996 {
997   "id":
998   "http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.WSS.json#",
999   "$schema": "http://json-schema.org/draft-04/schema#",
1000   "description" : "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
1001 reserved.",
1002   "title": "Weight Scale",
1003   "definitions": {
1004     "byte": {
1005       "type": "integer",
1006       "minimum": 0,
1007       "maximum": 255
1008     },
1009     "byteArray": {
1010       "type": "array",
1011       "items": { "$ref": "#/definitions/byte" },
1012       "minItems": 1,
1013       "uniqueItems": false
1014     },
1015     "org.bluetooth.characteristic.weight_measurement" : {
1016       "type": "object",
1017       "properties": {
1018         "weight_measurement[length - 3 : length - 1]" : {
1019           "$ref": "#/definitions/byteArray",
1020           "description": "Weight",
1021           "x-ocf-conversion": {
1022             "x-ocf-alias": "oic.r.weight",
1023             "x-to-ocf": [
1024               "length = len(weight_measurement)",
1025               "flags = weight_measurement[length - 1]",
1026               "timeoffset_len = 7 if (flags & 0x02) else 0",
1027               "oic.r.weight.weight = int.from_bytes(weight_measurement[length - 3 :
1028 length - 1], 'big')",
1029               "oic.r.weight.units = 'lb' if (flags & 0x01) else 'kg'"
1030             ],
1031             "x-from-ocf": [
1032               "N/A"
1033             ]
1034           }
1035         },
1036         "weight_measurement[length - 2 - userid_len - timeoffset_len - 3 : length -
1037 userid_len - timeoffset_len - 3]" : {
1038           "$ref": "#/definitions/byteArray",
1039           "description": "BMI",
1040           "x-ocf-conversion": {
1041             "x-ocf-alias": "oic.r.bmi",
1042             "x-to-ocf": [
1043               "length = len(weight_measurement)",
1044               "flags = weight_measurement[length - 1]",
1045               "timeoffset_len = 7 if (flags & 0x02) else 0",
1046               "userid_len = 1 if (flags & 0x04) else 0",
1047               "if (flags & 0x08):",
1048               "   oic.r.bmi.bmi = int.from_bytes(weight_measurement[length - 2 -
1049 userid_len - timeoffset_len - 3 : length - userid_len - timeoffset_len - 3], 'big')"
1050             ],
1051             "x-from-ocf": [
1052

```



```
1053         "N/A"
1054     ]
1055 }
1056 },
1057     "weight_measurement[length - height_len - userid_len - timeoffset_len - 3 :
1058 length - 2 - userid_len - timeoffset_len - 3]" : {
1059     "$ref": "#/definitions/byteArray",
1060     "description": "Height",
1061     "x-ocf-conversion": {
1062         "x-ocf-alias": "oic.r.height",
1063         "x-to-ocf": [
1064             "length = len(weight_measurement)",
1065             "flags = weight_measurement[length - 1]",
1066             "timeoffset_len = 7 if (flags & 0x02) else 0",
1067             "userid_len = 1 if (flags & 0x04) else 0",
1068             "height_len = 4 if (flags & 0x08) else 0",
1069             "if (flags & 0x08):",
1070             "    oic.r.height.height = int.from_bytes(weight_measurement[length -
1071 height_len - userid_len - timeoffset_len - 3 : length - 2 - userid_len - timeoffset_len
1072 - 3], 'big')",
1073             "    oic.r.height.units = 'in' if (flags & 0x01) else 'm'"
1074         ],
1075         "x-from-ocf": [
1076             "N/A"
1077         ]
1078     }
1079 }
1080 }
1081 },
1082 "org.bluetooth.characteristic.body_composition_measurement" : {
1083     "type": "object",
1084     "properties": {
1085         "body_composition_measurement[ length - 4 : length - 2]" : {
1086             "$ref": "#/definitions/byteArray",
1087             "description": "Body Fat Percentage",
1088             "x-ocf-conversion": {
1089                 "x-ocf-alias": "oic.r.body.fat",
1090                 "x-to-ocf": [
1091                     "length = len(body_composition_measurement)",
1092                     "oic.r.body.fat.bodyfat =
1093 int.from_bytes(body_composition_measurement[ length - 4 : length - 2], 'big')",
1094                     "oic.r.body.fat.units = '%"
1095                 ],
1096                 "x-from-ocf": [
1097                     "N/A"
1098                 ]
1099             }
1100         }
1101     },
1102     "body_composition_measurement[ length - bwm_len - slm_len - ffm_len - mm_len
1103 - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len - ffm_len -
1104 mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 ]" : {
1105         "$ref": "#/definitions/byteArray",
1106         "description": "Body Water Mass",
1107         "x-ocf-conversion": {
1108             "x-ocf-alias": "oic.r.body.water",
1109             "x-to-ocf": [
1110                 "length = len(body_composition_measurement)",
1111                 "flags_upperbyte = body_composition_measurement[length - 2]",
1112                 "flags_lowerbyte = body_composition_measurement[length - 1]",
1113                 "timestamp_len = 7 if (flags_lowerbyte & 0x02) else 0",
1114                 "userid_len = 1 if (flags_lowerbyte & 0x04) else 0",
1115                 "basal_len = 2 if (flags_lowerbyte & 0x08) else 0",
```



```
1116         "muscle_len = 2 if (flags_lowerbyte & 0x10) else 0",
1117         "mm_len = 2 if (flags_lowerbyte & 0x20) else 0",
1118         "ffm_len = 2 if (flags_lowerbyte & 0x40) else 0",
1119         "slm_len = 2 if (flags_lowerbyte & 0x80) else 0",
1120         "bwm_len = 2 if (flags_upperbyte & 0x01) else 0",
1121         "if (flags_lowerbyte & 0x01): ",
1122         "    oic.r.body.water.bwater =
1123 int.from_bytes(body_composition_measurement[ length - bwm_len - slm_len - ffm_len -
1124 mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len -
1125 ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 ], 'big')",
1126         "    oic.r.body.water.units = 'lb' if (flags_lowerbyte & 0x01) 'kg'"
1127     ],
1128     "x-from-ocf": [
1129         "N/A"
1130     ]
1131 }
1132 },
1133 "body_composition_measurement[ length - slm_len - ffm_len - mm_len -
1134 muscle_len - basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len -
1135 muscle_len - basal_len - userid_len - timestamp_len - 4 ]" : {
1136     "$ref": "#/definitions/byteArray",
1137     "description": "Soft Lean Mass",
1138     "x-ocf-conversion": {
1139         "x-ocf-alias": "oic.r.body.slm",
1140         "x-to-ocf": [
1141             "length = len(body_composition_measurement)",
1142             "flags_upperbyte = body_composition_measurement[length - 2]",
1143             "flags_lowerbyte = body_composition_measurement[length - 1]",
1144             "timestamp_len = 7 if (flags_lowerbyte & 0x02) else 0",
1145             "userid_len = 1 if (flags_lowerbyte & 0x04) else 0",
1146             "basal_len = 2 if (flags_lowerbyte & 0x08) else 0",
1147             "muscle_len = 2 if (flags_lowerbyte & 0x10) else 0",
1148             "mm_len = 2 if (flags_lowerbyte & 0x20) else 0",
1149             "ffm_len = 2 if (flags_lowerbyte & 0x40) else 0",
1150             "slm_len = 2 if (flags_lowerbyte & 0x80) else 0",
1151             "if (flags_lowerbyte & 0x01): ",
1152             "    oic.r.body.slm.bwater =
1153 int.from_bytes(body_composition_measurement[ length - slm_len - ffm_len - mm_len -
1154 muscle_len - basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len -
1155 muscle_len - basal_len - userid_len - timestamp_len - 4 ], 'big')",
1156             "    oic.r.body.slm.units = 'lb' if (flags_lowerbyte & 0x01) 'kg'"
1157         ],
1158         "x-from-ocf": [
1159             "N/A"
1160         ]
1161     }
1162 },
1163 "body_composition_measurement[ length - ffm_len - mm_len - muscle_len -
1164 basal_len - userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len -
1165 userid_len - timestamp_len - 4 ]" : {
1166     "$ref": "#/definitions/byteArray",
1167     "description": "Fat Free Mass",
1168     "x-ocf-conversion": {
1169         "x-ocf-alias": "oic.r.body.ffm",
1170         "x-to-ocf": [
1171             "length = len(body_composition_measurement)",
1172             "flags_upperbyte = body_composition_measurement[length - 2]",
1173             "flags_lowerbyte = body_composition_measurement[length - 1]",
1174             "timestamp_len = 7 if (flags_lowerbyte & 0x02) else 0",
1175             "userid_len = 1 if (flags_lowerbyte & 0x04) else 0",
1176             "basal_len = 2 if (flags_lowerbyte & 0x08) else 0",
1177             "muscle_len = 2 if (flags_lowerbyte & 0x10) else 0",
1178             "mm_len = 2 if (flags_lowerbyte & 0x20) else 0",
```



```
1179         "ffm_len = 2 if (flags_lowerbyte & 0x40) else 0",
1180         "if (flags_lowerbyte & 0x01): ",
1181         "     oic.r.body.ffm.bwater =
1182 int.from_bytes(body_composition_measurement[ length - ffm_len - mm_len - muscle_len -
1183 basal_len - userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len -
1184 userid_len - timestamp_len - 4 ], 'big')",
1185         "     oic.r.body.ffm.units = 'lb' if (flags_lowerbyte & 0x01) 'kg'"
1186     ],
1187     "x-from-ocf": [
1188         "N/A"
1189     ]
1190 }
1191 }
1192 }
1193 }
1194 },
1195
1196 "type": "object",
1197
1198 "allof": [
1199     { "$ref": "#/definitions/byte" },
1200     { "$ref": "#/definitions/byteArray" },
1201     { "$ref": "#/definitions/org.bluetooth.characteristic.weight_measurement" },
1202     { "$ref":
1203 "#/definitions/org.bluetooth.characteristic.body_composition_measurement" }
1204 ],
1205
1206 "required": [
1207     "weight_measurement[length - 3 : length - 1]"
1208 ]
1209 }
```