OCF 2.3 – BLE Mapping Spec – BTG

Legal Disclaimer

CONTENTS

67
68

Figures

No table of figures entries found.

Tables

1 Scope

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

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

2 Normative references

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

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

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

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

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

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

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

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

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

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

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

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

3 Terms, definitions and symbols

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

3.1 Terms and definitions

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

3.2 Symbols and abbreviations

None defined.

4 Document conventions and organization

4.1 Introduction

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

4.2 Conventions

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

4.3 Notation

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

Required (or shall or mandatory).

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

Recommended (or should).

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

Allowed (or allowed).

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

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

Conditionally required (CR)

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

DEPRECATED

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

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

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

Words that are emphasized are printed in *italic*.

4.4 Data types

Data types are defined in the OCF Core Specification.

5 Theory of Operation

5.1 Interworking Approach

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

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

Table 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**

5.2 Mapping Syntax

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

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

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

```
            "x-ocf-alias": "<corresponding OCF Resource type>",
            "x-to-ocf": [
               ...
               ...
            ],
            "x-from-ocf": [
               "N/A"
            ]
         }
      }
   }
}
```

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

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

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

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

6 Device Type Mapping

6.1 Introduction

This Section contains the mappings to Device Types.

6.2 BLE Profile to OCF Device Types

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

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

7 BLE Profile to Resource Equivalence

7.1 Introduction

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

7.2 BLE Services to OCF Resources

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

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

Device Information Service	oic.wk.p

8 Detailed Mappings

8.1 Blood Pressure Monitor Mapping

8.1.1 Introduction

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

8.1.2 Derived Model

org.bluetooth.characteristic.blood_pressure_measurement

8.1.3 Property Definition

Table 4 The property mapping for 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

8.1.4 Derived Model Definition

```
{
   "id":
"http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.BLP.json#",
   "$schema": "http://json-schema.org/draft-04/schema#",
   "description" : "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
reserved.",
   "title": "Blood Pressure",
   "definitions": {
      "byte": {
         "type": "integer",
```

```
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)",
```

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

```
            },
            "blood_pressure_measurement[length - 7 - timestamp_len - 2 : length - 7 -
timestamp_len]": {
               "$ref": "#/definitions/byteArray",
               "description": "Pulse Rate",
               "x-ocf-conversion": {
                  "x-ocf-alias": "oic.r.pulserate",
                  "x-to-ocf": [
                     "def ieee11073_Sfloat_2_Float(sfloat_value):",
                        "# reserved value for Infinity or NaN (Not a Number)",
                        "reserved_float_values = {",
                           "0x07FE:math.inf, # +INFINITY",
                           "0x07FF:math.nan, # NaN (Not a Number)",
                           "0x0800:math.nan, # NRes (Not at this Resolution)",
                           "0x0801:math.nan, # Reserved for future",
                           "0x0802:-math.inf # -INFINITY",
                        "}",
                        "mantissa = sfloat_value & 0x0FFF",
                        "exponent = sfloat_value >> 12",
                        "if (exponent >= 0x0008):",
                           "exponent = -((0x000F + 1) - exponent)",
                        "output = 0",
                        "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
                           "output = reserved_float_values[mantissa]",
                        "else:",
                           "if (mantissa >= 0x0800):",
                              "mantissa = -((0x0FFF + 1) - mantissa)",
                           "magnitude = pow(10.0, exponent)",
                           "output = (mantissa * magnitude)",
                        "return output",
                     "length = len(blood_pressure_measurement)",
                     "flags = blood_pressure_measurement[length - 1]",
                     "timestamp_len = 7 if (flags & 0x02) else 0",
                     "oic.r.pulserate.pulserate =
ieee11073_Sfloat_2_Float(blood_pressure_measurement[length - 7 - timestamp_len - 2 :
length - 7 - timestamp_len])"
                  ],
                  "x-from-ocf": [
                     "N/A"
                  ]
               }
            }
         }
      }
   },

   "type": "object",

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

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

8.2 Glucose Meter Mapping

8.2.1 Introduction

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

8.2.2 Derived Model

org.bluetooth.characteristic.glucose_measurement_context,
org.bluetooth.characteristic.glucose_measurement

8.2.3 Property Definition

Table 5 The property mapping for 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

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

glucose_measurement_context[length - 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 Property name	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

8.2.4 Derived Model Definition

```
{
   "id":
"http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.GLP.json#",
   "$schema": "http://json-schema.org/draft-04/schema#",
   "description" : "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
reserved.",
   "title": "Glucose",
   "definitions": {
      "byte": {
         "type": "integer",
         "minimum": 0,
         "maximum": 255
      },
      "byteArray": {
         "type": "array",
         "items": { "$ref": "#/definitions/byte" },
         "minItems": 1,
         "uniqueItems": false
      },
      "org.bluetooth.characteristic.glucose_measurement": {
         "type": "object",
         "properties": {
            "glucose_measurement[length - 2 - timeoffset_len - 10 : length -
timeoffset_len - 10]": {
               "$ref": "#/definitions/byteArray",
               "description": "Glucose Concentration",
               "x-ocf-conversion": {
                  "x-ocf-alias": "oic.r.glucose",
                  "x-to-ocf": [
                     "def ieee11073_Sfloat_2_Float(sfloat_value):",
                        "# reserved value for Infinity or NaN (Not a Number)",
                        "reserved_float_values = {",
                           "0x07FE:math.inf, # +INFINITY",
                           "0x07FF:math.nan, # NaN (Not a Number)",
                           "0x0800:math.nan, # NRes (Not at this Resolution)",
                           "0x0801:math.nan, # Reserved for future",
                           "0x0802:-math.inf # -INFINITY",
                        "}",
                        "mantissa = sfloat_value & 0x0FFF",
                        "exponent = sfloat_value >> 12",
                        "if (exponent >= 0x0008):",
                           "exponent = -((0x000F + 1) - exponent)",
                        "output = 0",
                        "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
                           "output = reserved_float_values[mantissa]",
                        "else:",
                           "if (mantissa >= 0x0800):",
                              "mantissa = -((0x0FFF + 1) - mantissa)",
                           "magnitude = pow(10.0, exponent)",
                           "output = (mantissa * magnitude)",
                        "return output",
                     "length = len(glucose_measurement)",
                     "flags = glucose_measurement[length - 1]",
                     "timeoffset_len = 2 if (flags & 0x01) else 0",
                     "if (flags & 0x02) == True:",
                     "   glucose = ieee11073_Sfloat_2_Float(glucose_measurement[length - 2
- timeoffset_len - 10 : length - timeoffset_len - 10])",
                     "   oic.r.glucose.glucose = (glucose * 1000) if (flags & 0x04) else
(glucose * 0.1 * 1000 * 1000)",
                     "   oic.r.glucose.units = 'mmol/L' if (flags & 0x04) else 'mg/dL'",
```

```
                     "if (flags & 0x02) == False:",
                     "   oic.r.glucose.glucose = 0",
                     "   oic.r.glucose.units = 'mmol/L'"
                  ],
                  "x-from-ocf": [
                     "N/A"
                  ]
               }
            },
            "glucose_measurement[length - 1 - 2 - timeoffset_len - 10]": {
               "$ref": "#/definitions/byteArray",
               "description": "Sample Location",
               "x-ocf-conversion": {
                  "x-ocf-alias": "oic.r.glucose.samplelocation",
                  "x-to-ocf": [
                     "length = len(glucose_measurement)",
                     "flags = glucose_measurement[length - 1]",
                     "timeoffset_len = 2 if (flags & 0x01) else 0",
                     "if (flags & 0x02):",
                     "   samplelocation = { 1:'finger', 2:'ast', 3:'earlobe',
4:'ctrlsolution' }",
                     "   oic.r.glucose.samplelocation.samplelocation =
samplelocation[glucose_measurement[length - 1 - 2 - timeoffset_len - 10] & 0xf0]"
                  ],
                  "x-from-ocf": [
                     "N/A"
                  ]
               }
            }
         }
      },

      "org.bluetooth.characteristic.glucose_measurement_context": {
         "type": "object",
         "properties": {
            "glucose_measurement_context[length - carb_len - extflags_len - 3 : length -
1 - extflags_len - 3]": {
               "$ref": "#/definitions/byteArray",
               "description": "Carbohydrate",
               "x-ocf-conversion": {
                  "x-ocf-alias": "oic.r.glucose.carb",
                  "x-to-ocf": [
                     "def ieee11073_Sfloat_2_Float(sfloat_value):",
                        "# reserved value for Infinity or NaN (Not a Number)",
                        "reserved_float_values = {",
                           "0x07FE:math.inf, # +INFINITY",
                           "0x07FF:math.nan, # NaN (Not a Number)",
                           "0x0800:math.nan, # NRes (Not at this Resolution)",
                           "0x0801:math.nan, # Reserved for future",
                           "0x0802:-math.inf # -INFINITY",
                        "}",
                        "mantissa = sfloat_value & 0x0FFF",
                        "exponent = sfloat_value >> 12",
                        "if (exponent >= 0x0008):",
                           "exponent = -((0x000F + 1) - exponent)",
                        "output = 0",
                        "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
                           "output = reserved_float_values[mantissa]",
                        "else:",
                           "if (mantissa >= 0x0800):",
                              "mantissa = -((0x0FFF + 1) - mantissa)",
                           "magnitude = pow(10.0, exponent)",
                           "output = (mantissa * magnitude)",
```

```
                     "return output",
                  "length = len(glucose_measurement_context)",
                  "flags = glucose_measurement_context[length - 1]",
                  "extflags_len = 1 if (flags & 0x80) else 0",
                  "carb_len = 3 if (flags & 0x01) else 0",
                  "if (flags & 0x01): ",
                  "   oic.r.glucose.carb.carb =
ieee11073_Sfloat_2_Float(glucose_measurement_context[length - carb_len - extflags_len -
3 : length - 1 - extflags_len - 3]) * 1000"
               ],
               "x-from-ocf": [
                  "N/A"
               ]
            }
         },
         "glucose_measurement_context[length - 1 - extflags_len - 3]": {
            "$ref": "#/definitions/byteArray",
            "description": "Carbohydrate ID",
            "x-ocf-conversion": {
               "x-ocf-alias": "oic.r.glucose.carb",
               "x-to-ocf": [
                  "length = len(glucose_measurement_context)",
                  "flags = glucose_measurement_context[length - 1]",
                  "extflags_len = 1 if (flags & 0x80) else 0",
                  "if (flags & 0x01): ",
                  "   meal = { 1:'breakfast', 2:'lunch', 3:'dinner', 4:'snack',
5:'drink', 6:'supper', 7:'brunch' }",
                  "   oic.r.glucose.carb.meal = meal[glucose_measurement_context[length
- 1 - extflags_len - 3]]"
               ],
               "x-from-ocf": [
                  "N/A"
               ]
            }
         },
         "glucose_measurement_context[length - 2 - health_len - meal_len - carb_len -
extflags_len - 3]": {
            "$ref": "#/definitions/byteArray",
            "description": "Exercise Intensity",
            "x-ocf-conversion": {
               "x-ocf-alias": "oic.r.glucose.exercise",
               "x-to-ocf": [
                  "length = len(glucose_measurement_context)",
                  "flags = glucose_measurement_context[length - 1]",
                  "extflags_len = 1 if (flags & 0x80) else 0",
                  "carb_len = 3 if (flags & 0x01) else 0",
                  "meal_len = 1 if (flags & 0x02) else 0",
                  "health_len = 1 if (flags & 0x04) else 0",
                  "if (flags & 0x08): ",
                  "   oic.r.glucose.exercise.exercise =
glucose_measurement_context[length - 2 - health_len - meal_len - carb_len - extflags_len
- 3]"
               ],
               "x-from-ocf": [
                  "N/A"
               ]
            }
         },
         "glucose_measurement_context[length - hba1c_len - medication_len -
exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length -
medication_len - exercise_len - health_len - meal_len - carb_len - extflags_len - 3]": {
            "$ref": "#/definitions/byteArray",
            "description": "HbA1c",
```

```
654                "x-ocf-conversion": {
655                   "x-ocf-alias": "oic.r.glucose.hba1c",
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                      "hba1c_len = 2 if (flags & 0x40) else 0",
688                      "if (flags & 0x40):",
689                      "   oic.r.glucose.hba1c.hba1c =
690   ieee11073_Sfloat_2_Float(glucose_measurement_context[length - hba1c_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' }",
```

```
                     "   oic.r.glucose.health.health =
health[glucose_measurement_context[length - health_len - meal_len - carb_len -
extflags_len - 3] & 0xf0]"
                  ],
                  "x-from-ocf": [
                     "N/A"
                  ]
               }
            },
            "glucose_measurement_context[length - health_len - meal_len - carb_len -
extflags_len - 3]": {
               "$ref": "#/definitions/byteArray",
               "description": "Tester",
               "x-ocf-conversion": {
                  "x-ocf-alias": "oic.r.glucose.tester",
                  "x-to-ocf": [
                     "length = len(glucose_measurement_context)",
                     "flags = glucose_measurement_context[length - 1]",
                     "extflags_len = 1 if (flags & 0x80) else 0",
                     "carb_len = 3 if (flags & 0x01) else 0",
                     "meal_len = 1 if (flags & 0x02) else 0",
                     "health_len = 1 if (flags & 0x04) else 0",
                     "if (flags & 0x04): ",
                     "   tester = { 1:'self', 2:'hcp', 3:'lab' }",
                     "   oic.r.glucose.tester.tester =
tester[glucose_measurement_context[length - health_len - meal_len - carb_len -
extflags_len - 3] & 0x0f]"
                  ],
                  "x-from-ocf": [
                     "N/A"
                  ]
               }
            },
            "glucose_measurement_context[length - meal_len - carb_len - extflags_len -
3]": {
               "$ref": "#/definitions/byteArray",
               "description": "Meal",
               "x-ocf-conversion": {
                  "x-ocf-alias": "oic.r.glucose.meal",
                  "x-to-ocf": [
                     "length = len(glucose_measurement_context)",
                     "flags = glucose_measurement_context[length - 1]",
                     "extflags_len = 1 if (flags & 0x80) else 0",
                     "carb_len = 3 if (flags & 0x01) else 0",
                     "meal_len = 1 if (flags & 0x02) else 0",
                     "if (flags & 0x02): ",
                     "   meal = { 1:'preprandial', 2:'postprandial', 3:'fasting',
4:'casual', 5:'bedtime' }",
                     "   oic.r.glucose.meal.meal = meal[glucose_measurement_context[length
- meal_len - carb_len - extflags_len - 3]]"
                  ],
                  "x-from-ocf": [
                     "N/A"
                  ]
               }
            },
            "glucose_measurement_context[length - medication_len - exercise_len -
health_len - meal_len - carb_len - extflags_len - 3 : length - 1 - exercise_len -
health_len - meal_len - carb_len - extflags_len - 3]": {
               "$ref": "#/definitions/byteArray",
               "description": "Medication",
               "x-ocf-conversion": {
                  "x-ocf-alias": "oic.r.glucose.medication",
```

```
                  "x-to-ocf": [
                     "def ieee11073_Sfloat_2_Float(sfloat_value):",
                        "# reserved value for Infinity or NaN (Not a Number)",
                        "reserved_float_values = {",
                           "0x07FE:math.inf, # +INFINITY",
                           "0x07FF:math.nan, # NaN (Not a Number)",
                           "0x0800:math.nan, # NRes (Not at this Resolution)",
                           "0x0801:math.nan, # Reserved for future",
                           "0x0802:-math.inf # -INFINITY",
                        "}",
                        "mantissa = sfloat_value & 0x0FFF",
                        "exponent = sfloat_value >> 12",
                        "if (exponent >= 0x0008):",
                           "exponent = -((0x000F + 1) - exponent)",
                        "output = 0",
                        "if (mantissa >= 0x07FE and mantissa <= 0x0802):",
                           "output = reserved_float_values[mantissa]",
                        "else:",
                           "if (mantissa >= 0x0800):",
                              "mantissa = -((0x0FFF + 1) - mantissa)",
                           "magnitude = pow(10.0, exponent)",
                           "output = (mantissa * magnitude)",
                        "return output",
                     "length = len(glucose_measurement_context)",
                     "flags = glucose_measurement_context[length - 1]",
                     "extflags_len = 1 if (flags & 0x80) else 0",
                     "carb_len = 3 if (flags & 0x01) else 0",
                     "meal_len = 1 if (flags & 0x02) else 0",
                     "health_len = 1 if (flags & 0x04) else 0",
                     "exercise_len = 3 if (flags & 0x08) else 0",
                     "medication_len = 3 if (flags & 0x10) else 0",
                     "hba1c_len = 2 if (flags & 0x40) else 0",
                     "if (flags & 0x10): ",
                     "   medication =
ieee11073_Sfloat_2_Float(glucose_measurement_context[length - medication_len -
exercise_len - health_len - meal_len - carb_len - extflags_len - 3 : length - 1 -
exercise_len - health_len - meal_len - carb_len - extflags_len - 3])",
                     "   oic.r.glucose.medication.medication = medication * 1000",
                     "   oic.r.glucose.medication.units = 'mL' if (flags & 0x20) else
'mg'"
                  ],
                  "x-from-ocf": [
                     "N/A"
                  ]
               }
            },
            "glucose_measurement_context[length - 1 - exercise_len - health_len -
meal_len - carb_len - extflags_len - 3]": {
               "$ref": "#/definitions/byteArray",
               "description": "Medication ID",
               "x-ocf-conversion": {
                  "x-ocf-alias": "oic.r.glucose.medication",
                  "x-to-ocf": [
                     "length = len(glucose_measurement_context)",
                     "flags = glucose_measurement_context[length - 1]",
                     "extflags_len = 1 if (flags & 0x80) else 0",
                     "carb_len = 3 if (flags & 0x01) else 0",
                     "meal_len = 1 if (flags & 0x02) else 0",
                     "health_len = 1 if (flags & 0x04) else 0",
                     "exercise_len = 3 if (flags & 0x08) else 0",
                     "if (flags & 0x10): ",
                     "   regimen = { 1:'rapidacting', 2:'shortacting',
3:'intermediateacting', 4:'longacting', 5:'premix' }",
```

```
                    "   oic.r.glucose.medication.regimen =
regimen[ glucose_measurement_context[length - 1 - exercise_len - health_len - meal_len -
carb_len - extflags_len - 3] ]"
                  ],
                  "x-from-ocf": [
                     "N/A"
                  ]
               }
            }
         }
      }
   },

   "type": "object",

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

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

8.3 Body Thermometer Mapping

8.3.1 Introduction

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

8.3.2 Derived Model

org.bluetooth.characteristic.temperature_measurement.

8.3.3 Property Definition

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

temperature_measurement[length - temperaturetype_len - timestamp_len - 5]	oic.r.body.location.temperature	length = len(temperature_measurement) flags = temperature_measurement[length - 1] timestamp_len = 7 if (flags & 0x02) 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]]	N/A

8.3.4 Derived Model Definition

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

```
                              "mantissa = -((0x00FFFFFF + 1) - mantissa)",
                           "magnitude = pow(10.0, exponent)",
                           "output = (mantissa * magnitude)",
                        "return output",
                     "length = len(temperature_measurement)",
                     "flags = temperature_measurement[length - 1]",
                     "oic.r.temperature.temperature =
ieee11073_Float_2_Float(temperature_measurement[ length - 5 : length - 1 ])",
                     "oic.r.temperature.units = 'F' if (flags & 0x01) else 'C'"
                  ],
                  "x-from-ocf": [
                     "N/A"
                  ]
               }
            },
            "temperature_measurement[ length - temperaturetype_len - timestamp_len -
5 ]" : {
               "$ref": "#/definitions/byteArray",
               "description": "Temperature Type",
               "x-ocf-conversion": {
                  "x-ocf-alias": "oic.r.body.location.temperature",
                  "x-to-ocf": [
                     "length = len(temperature_measurement)",
                     "flags = temperature_measurement[length - 1]",
                     "timestamp_len = 7 if (flags & 0x02) 0",
                     "temperaturetype_len = 1 if (flags & 0x04) 0",
                     "if (flags & 0x04):",
                     "   bloc = { 1:'xxx', 2:'body', 3:'ear', 4:'finger', 5:'gastro',
6:'mouth', 7:'rectum', 8:'toe', 9:'tympanum' }",
                     "   oic.r.body.location.temperature.bloc =
bloc[temperature_measurement[ length - temperaturetype_len - timestamp_len - 5 ] ]"
                  ],
                  "x-from-ocf": [
                     "N/A"
                  ]
               }
            }
         }
      },

      "type": "object",

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

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

8.4 Body Scale Mapping

8.4.1 Introduction

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

8.4.2 Derived Model

org.bluetooth.characteristic.weight_measurement.
org.bluetooth.characteristic.body_composition_measurement.

8.4.3 Property Definition

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

Table 9 The property mapping for 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

		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'	
body_composition_measurement[length - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4]	oic.r.body.ffm	length = len(body_composition_measurement) flags_upperbyte = body_composition_measurement[length - 2] flags_lowerbyte = body_composition_measurement[length - 1] timestamp_len = 7 if (flags_lowerbyte & 0x02) else 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'	N/A
body_composition_measurement[length - slm_len - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 : length - ffm_len - mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4]	oic.r.body.slm	length = len(body_composition_measurement) flags_upperbyte = body_composition_measurement[length - 2] flags_lowerbyte = body_composition_measurement[length - 1] timestamp_len = 7 if (flags_lowerbyte & 0x02) else 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'	N/A
body_composition_measurement[length - 4 : length - 2]	oic.r.body.fat	length = len(body_composition_measurement)	N/A

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

8.4.4 Derived Model Definition

```
{
   "id":
"http://openinterconnect.org/bleocfmapping/schemas/org.bluetooth.profile.WSS.json#",
   "$schema": "http://json-schema.org/draft-04/schema#",
   "description" : "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights
reserved.",
   "title": "Weight Scale",
   "definitions": {
      "byte": {
         "type": "integer",
         "minimum": 0,
         "maximum": 255
      },
      "byteArray": {
         "type": "array",
         "items": { "$ref": "#/definitions/byte" },
         "minItems": 1,
         "uniqueItems": false
      },
      "org.bluetooth.characteristic.weight_measurement" : {
         "type": "object",
         "properties": {
            "weight_measurement[length - 3 : length - 1]" : {
               "$ref": "#/definitions/byteArray",
               "description": "Weight",
               "x-ocf-conversion": {
                  "x-ocf-alias": "oic.r.weight",
                  "x-to-ocf": [
                     "length = len(weight_measurement)",
                     "flags = weight_measurement[length - 1]",
                     "timeoffset_len = 7 if (flags & 0x02) else 0",
                     "oic.r.weight.weight = int.from_bytes(weight_measurement[length - 3 :
length - 1], 'big')",
                     "oic.r.weight.units = 'lb' if (flags & 0x01) else 'kg'"
                  ],
                  "x-from-ocf": [
                     "N/A"
                  ]
               }
            },
            "weight_measurement[length - 2 - userid_len - timeoffset_len - 3 : length -
userid_len - timeoffset_len - 3]" : {
               "$ref": "#/definitions/byteArray",
               "description": "BMI",
               "x-ocf-conversion": {
                  "x-ocf-alias": "oic.r.bmi",
                  "x-to-ocf": [
                     "length = len(weight_measurement)",
                     "flags = weight_measurement[length - 1]",
                     "timeoffset_len = 7 if (flags & 0x02) else 0",
                     "userid_len = 1 if (flags & 0x04) else 0",
                     "if (flags & 0x08):",
                     "   oic.r.bmi.bmi = int.from_bytes(weight_measurement[length - 2 -
userid_len - timeoffset_len - 3 : length - userid_len - timeoffset_len - 3], 'big')"
                  ],
                  "x-from-ocf": [
```

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

      "org.bluetooth.characteristic.body_composition_measurement" : {
         "type": "object",
         "properties": {
            "body_composition_measurement[ length - 4 : length - 2]" : {
               "$ref": "#/definitions/byteArray",
               "description": "Body Fat Percentage",
               "x-ocf-conversion": {
                  "x-ocf-alias": "oic.r.body.fat",
                  "x-to-ocf": [
                     "length = len(body_composition_measurement)",
                     "oic.r.body.fat.bodyfat =
int.from_bytes(body_composition_measurement[ length - 4 : length - 2], 'big')",
                     "oic.r.body.fat.units = '%'"
                  ],
                  "x-from-ocf": [
                     "N/A"
                  ]
               }
            },
            "body_composition_measurement[ length - bwm_len - slm_len - ffm_len - mm_len
- muscle_len - basal_len - userid_len - timestamp_len - 4 : length - slm_len - ffm_len -
mm_len - muscle_len - basal_len - userid_len - timestamp_len - 4 ]" : {
               "$ref": "#/definitions/byteArray",
               "description": "Body Water Mass",
               "x-ocf-conversion": {
                  "x-ocf-alias": "oic.r.body.water",
                  "x-to-ocf": [
                     "length = len(body_composition_measurement)",
                     "flags_upperbyte = body_composition_measurement[length - 2]",
                     "flags_lowerbyte = body_composition_measurement[length - 1]",
                     "timestamp_len = 7 if (flags_lowerbyte & 0x02) else 0",
                     "userid_len = 1 if (flags_lowerbyte & 0x04) else 0",
                     "basal_len = 2 if (flags_lowerbyte & 0x08) else 0",
```

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

```
                    "ffm_len = 2 if (flags_lowerbyte & 0x40) else 0",
                    "if (flags_lowerbyte & 0x01): ",
                    "   oic.r.body.ffm.bwater =
int.from_bytes(body_composition_measurement[ length - ffm_len - mm_len - muscle_len -
basal_len - userid_len - timestamp_len - 4 : length - mm_len - muscle_len - basal_len -
userid_len - timestamp_len - 4 ], 'big')",
                    "   oic.r.body.ffm.units = 'lb' if (flags_lowerbyte & 0x01) 'kg'"
                ],
                "x-from-ocf": [
                   "N/A"
                ]
             }
          }
       }
    },

    "type": "object",

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

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