Device WebAPI-PCH

Candidate Version 1.0 – 19 Apr 2016

Open Mobile Alliance

OMA-ER-Device_WebAPIs-V1_0-20160419-C

Contents

Figures

Tables

1. Scope

This Enabler Release (ER) document is a combined document that includes requirements, architecture and technical specification of the Device WebAPIs Enabler.

The scope of OMA Device WebAPI enabler will include:

- Requirements, architecture and specifications for web-based APIs to expose services available from external devices and internal enablers through Extension Plug-Ins to applications.
- The web-based APIs that will work in the framework that GotAPI (Generic Open Terminal API Framework) defines, where the web-based APIs are implemented in the Extension Plug-Ins and exposing the services from the external devices or internal enablers that are connected with the Extension Plug-Ins.
- The framework provided by the combination of GotAPI and Device WebAPI to enable applications to work through standardized APIs with external devices or internal enablers, as GotAPI itself does not standardize the APIs to be implemented in the Extension Plug-Ins.
- Web-based APIs that will initially address such areas as healthcare devices, DWAPI-PCH (Personal Connected Healthcare) and other areas where standardization will help solving application interoperability problems.

OMA will continue expanding the coverage of the standaridized Device WebAPIs in areas where standardization helps the markets to expand and innovate.

2. References

2.1 Normative References

[EventSource] “Server-Sent Events”, Worldwide Web Consortium (W3C), URL:http://dev.w3.org/html5/eventsource/ (latest working draft)

[GotAPI 1.1] Generic Open Terminal API Framework (GotAPI), Candidate Version 1.1 – 15 Dec 2015 URL:http://www.openmobilealliance.org/

[HTTP/1.1] “Hypertext Transfer Protocol -- HTTP/1.1”, Internet Engineering Task Force (IETF), URL:http://tools.ietf.org/search/rfc2616

[HTTP/2.0] “Hypertext Transfer Protocol version 2.0”, Internet Engineering Task Force (IETF), URL:http://tools.ietf.org/search/draft-ietf-httpbis-http2-09 (latest working draft)

[JSON-RPC] “JSON-RPC 2.0 Specification”, JSON-RPC Working Group, URL:http://www.jsonrpc.org/specification

[RFC2119] “Key words for use in RFCs to Indicate Requirement Levels”, S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt

[SCRRULES] “SCR Rules and Procedures”, Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/

[WebSocket] “The WebSocket API, Worldwide Web Consortium (W3C), URL:http://dev.w3.org/html5/websockets/ (latest working draft)

2.2 Informative References

[CSEA] “Client Side Enabler API (CSEA)”, Version 1.0, Open Mobile Alliance™, OMA-RRP-CSEA-V1_0, URL:http://www.openmobilealliance.org/

[OMADICT] “Dictionary for OMA Specifications”, Version 2.9, Open Mobile Alliance™, OMA-ORG-Dictionary-V2.9, URL:http://www.openmobilealliance.org/

[OMNA] "OMA Naming Authority". Open Mobile Alliance™. URL:http://www.openmobilealliance.org/tech/omna.aspx

[WRAPI] “Web Runtime API (WRAPI”, Version 1.0, Open Mobile Alliance™, OMA-ERP-WRAPI-V1_0, URL:http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

Agent	A node that collects and transmits personal health data to an associated manager.
API Patterns	Design guidelines and requirements for definition of APIs
Browser Context	Web applications executing under a Web browser as Web runtime environment.
Datagram	An API providing access to UDP protocol based networking.
Device	A physical device implementing either an agent or manager role.
ECMAScript	Use definition from [OMADICT].
Hybrid Native/Web App	An application designed to execute under the native OS / middleware environment of a device, and that use native APIs for the execution of web content in addition to native code.
JavaScript	Use definition from [OMADICT].
Manager	A node receiving data from one or more agent systems. Examples of managers include a cellular phone, health appliance, set top box, or computer system.
Native App	An application designed to execute under the native OS / middleware environment of a device.
Personal Health Device	A device used in personal health applications.
Socket	An API providing access to TCP protocol based networking.
Uniform Resource Identifier	Use definition from [OMADICT].
User Agent	Use definition from [OMADICT].
Web	The World Wide Web, a content and application framework based upon hypertext and related technologies, e.g. XML, JavaScript/ECMAScript, CSS, etc.
Web Application	An application designed using Web technologies (e.g. HTML, CSS, and Javascript).
Web IDL	An IDL language for Web application APIs
Web Runtime Application	A client-side Web application that is executed in Web runtime environments.
Web Runtime Environment	Client software that supports the execution of Web applications (e.g. browsers or widget engines).
WebSocket	An API providing networking services per the WebSocket standard [WebSocket].
Widget Context	Web applications installed and executing under a W3C Widget [W3C-Widgets] engine as Web runtime environment.
Widget Engine	Software which supports the execution of Web applications running outside a browser context, e.g. with the same functional capabilities as browsers but without the user interface functions provided by a browser, including window frames, menus, toolbars and scroll bars.

3.3 Abbreviations

API	Application Programming Interface
EventSource	The EventSource API

HTTP	HyperText Transfer Protocol
IDL	Interface Definition Language
JSON	JavaScript Object Notation
MIME	Multipurpose Internet Mail Extensions
OMA	Open Mobile Alliance
REST	REpresentational State Transfer
RPC	Remote Procedure Call
SCR	Static Conformance Requirements
TS	Technical Specification
UA	User Agent
UE	User Equipment
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WRAPI	The OMA Web Runtime API enabler
XML	eXtensible Markup Language
XSD	XML Schema Definition

4. Introduction

External devices that are connected with smartphones are increasingly gaining mainstream acceptance and we are starting to see rapid adoption of such devices.

While there are various types of new devices and sensors to be connected with smartphones coming out, there are fundamental issue to be solved for certain markets:

- Since there are no open standardized APIs and frameworks that application developers can use for the same type of devices, developers are required to customize their applications for each and every different device.
- In order to access features from applications, some environments mandate that the users' data must be routed through certain entities, e.g., servers outside user's control. As such, it is difficult to ensure data confidentiality and privacy to such a level where certain vertical markets require.

As the first step to solve this problem, OMA has standardized GotAPI (Generic Open Terminal API Framework) [***]. GotAPI provides the framework to enable applications (native, hybrid and web applications) to work with external devices and internal enablers through GotAPI Servers and Extension Plug-Ins based on web technologies. There are multiple Extension Plug-Ins to be expected and each Extension Plug-In is connected to external devices and internal enablers. Each Extension Plug-Ins implements web-based APIs to expose services (or data) from those connected. The applications securely access the web-based APIs under the framework that GotAPI provides. The figure-1 shows the overview of GotAPI's framework.

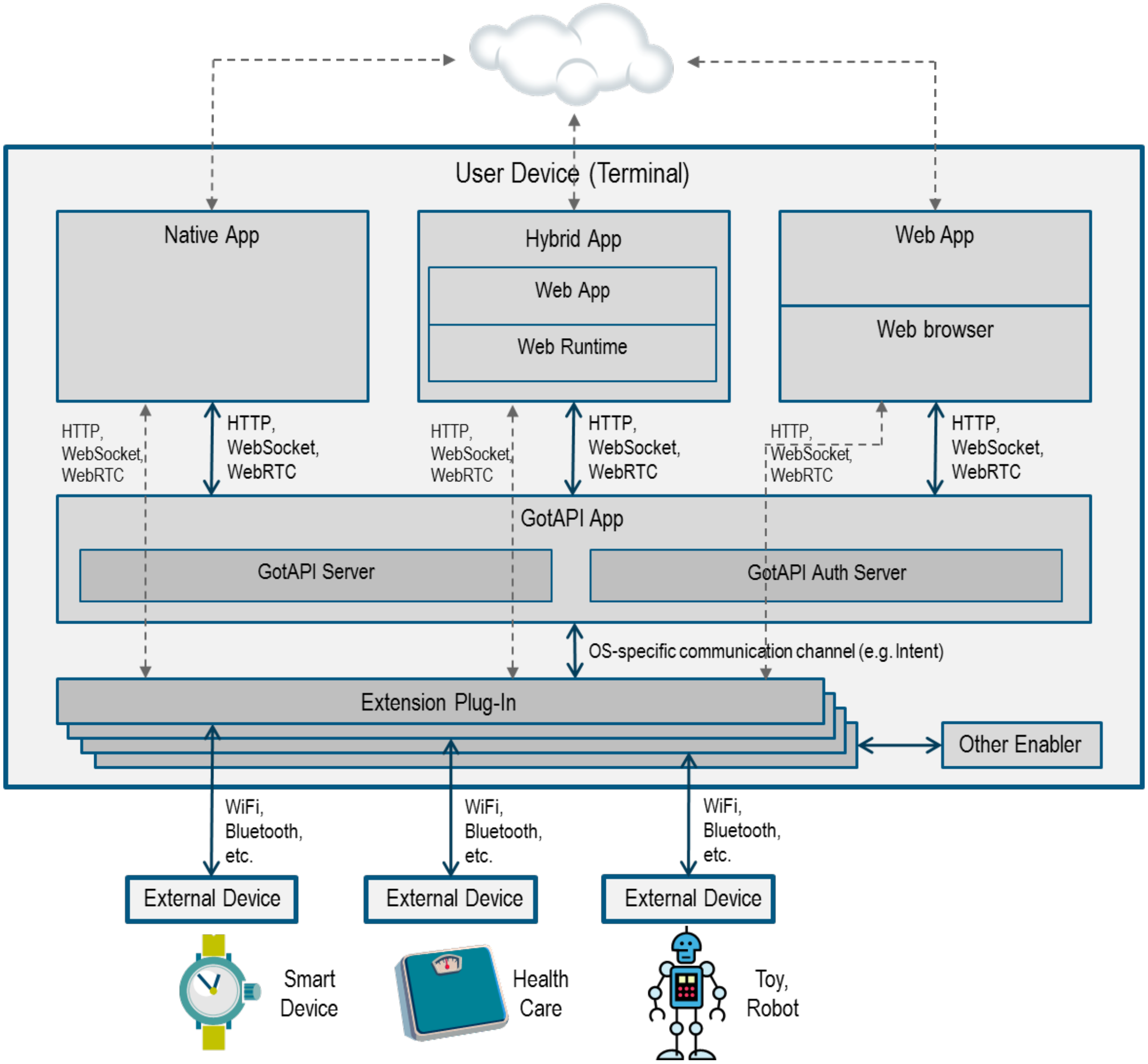


Figure 1 Overview of GotAPI's framework

GotAPI itself does not standardize the web-based APIs that are implemented in the Extension Plug-Ins, and it is left open for implementers of each Extension Plug-In. This openness enables many external device vendors to freely provide unique and differentiated new services through the GotAPI framework.

On the other hand, for certain markets, standardizing the web-based APIs is desired. Standardized web-based API will enable open markets for new applications by 3rd party developers that will rapidly innovat and grow the market, while ensuring the interoperability and security.

This specification specifies OMA Device WebAPIs Enabler. In contrast to OMA GotAPI being a versatile web application framework, OMA Device WebAPIs Enabler specifies web-based APIs for certain types of external devices or internal services to work consistently in the GotAPI framework. It enables applications to access specific types of external devices or internal services. OMA Device WebAPIs Enabler will offer a series of specifications to address different types of devices or internal services where standardization is needed.

DWAPI-PCH:

OMA has identified personal connected healthcare market is looking for standardized APIs, and OMA Device WebAPIs Enabler addresses this issue in order to ensure service interoperability between the same type of devices, as an alternative to siloed and non-interoperable devices.

For personal connected healthcare devices, the IEEE 11073 family of standard defines a large number of healthcare devices. OMA Device WebAPIs Enabler will develop web-based APIs to expose services from devices that are based on IEEE 11073 standards, where the APIs will be implemented in Extension Plug-Ins under the GotAPI framework. These API specifications are called as DWAPI-PCH, which stands for Device WebAPI for Personal Connected Healthcare. It is one of OMA Device WebAPIs Enabler specifications and specifies web-based APIs for different types of IEEE 11073 devices in a series of specifications.

4.1 IEEE 11073 Family of Standards Overview

The ISO/IEEE 11073 family of standards is based on an object-oriented systems management paradigm. Data (measurement, state, and so on) are modeled in the form of information objects that are accessed and manipulated using an object access service protocol.

The ISO/IEEE 11073-20601 Data Exchange Protocol (known as 20601) provides a framework for information and modelling, information access and measurement data transfer suitable to a wide variety of personal health devices. Examples of such health devices are as follows: weighing scales, thermometers, pulse oximeters, blood pressure monitors, and glucose meters. In addition to health and fitness sensors, the protocol is designed to support a range of home health sensors. This enables interoperability between a data management device to process, display or transfer the specific measurements.

20601 core protocol specification, which describe the tools to represent and convey data, and 104xx Device Data Specialization specifications, which provide details on how the 20601 tools are applied for each health device's implementation.

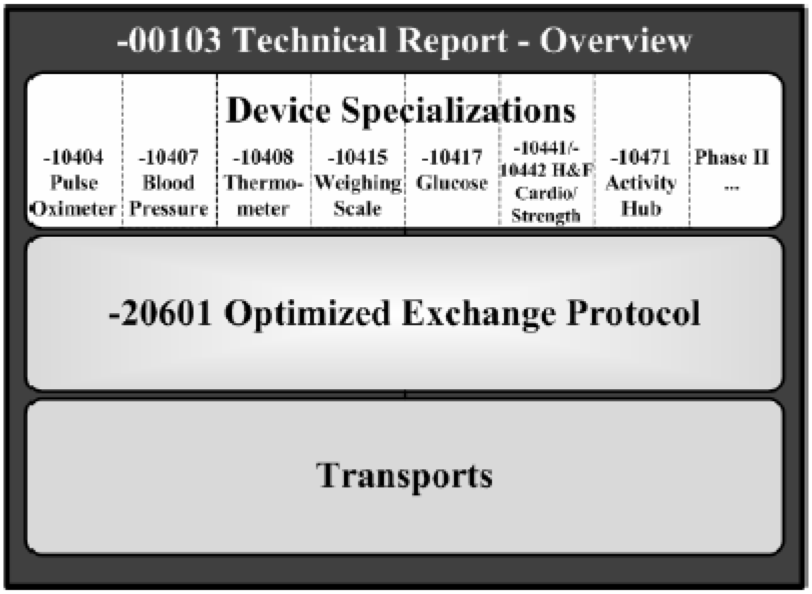


Figure 2 IEEE 11073 Overview

4.2 Version 1.0

Device WebAPIs version 1.0 includes the functionality:

- Requirements and API specifications for DWAPI-PCH, with selected device classes from IEEE 11073 based on market requirements, based on the GotAPI 1.1 framework
- Supporting assets

5. Device WebAPIs Enabler release description (Informative)

This release focuses on the functions of exposing the data from external devices to applications. The Device WebAPIs enabler will utilize the GotAPIs enabler and specify the APIs under the GotAPIs framework.

6. Requirements (Normative)

Edtor’s Note: Most of the security requirements should have already been defined in GotAPI 1.1.

If there is anything that is missing or unaddressed, we need to articulate them in thie security section.

6.1 High-Level Functional Requirements: GotAPI Adherence

In contrast to OMA GotAPI being a versatile web application framework, OMA Device WebAPIs Enabler specifies web-based APIs for certain types of external devices or internal enablers, to work consistently in the GotAPI framework. Therefore, the OMA Device WebAPIs Enabler must adhere to all the GotAPI 1.1 specifications, including, but not limited to, those of Extension Plug-Ins, data formats, sequence flows, security measures and considerations. Where necessary to ensure interoperability, OMA Device WebAPIs Enabler may standardize specific values or data for certain types of devices consistently within the GotAPI framework.

Label	Description	Release
HD-GOT-01	Device WebAPIs Enabler SHALL adhere to all the GotAPI 1.1 specifications [GotAPI 1.1], including, but not limited to, those of Extension Plug-Ins, data formats, sequence flows, security measures and considerations.	1.0
HD- GOT -02	Device WebAPIs Enabler MAY define specific values or data for certain types of devices that will work consistently within the GotAPI 1.1 framework with other types of devices and applications.	1.0
HD- GOT -03	The Plug-In SHALL be compliant to the GotAPI Extension Plug-Ins as specified in the GotAPI 1.1 specification.	1.0
HD- GOT -04	The framework hosting Plug-Ins SHALL support the security requirements defined in GotAPI 1.1.	1.0

6.2 High-Level Functional Requirements: DWAPI-PCH

The Plug-Ins and the APIs designed for consumer/ personal use perspective. The following requirements specify the guidelines for ***all*** Health Device Plug-Ins. Values, when reported, are reported as Strings or MDER FLOATs.

MDER FLOATs are used to report integers or real numbers. The reason for using MDER FLOATs is to capture precision as reported by the device. An MDER FLOAT is a 32 bit integer interpreted as follows:

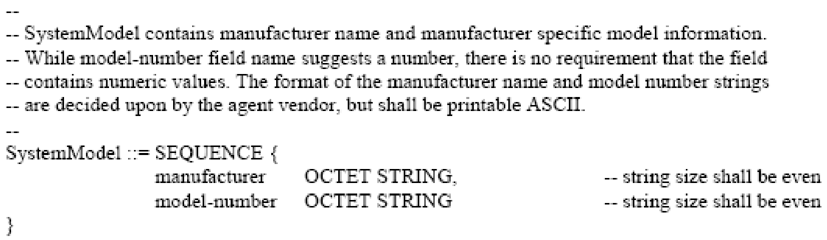
- The most significant 8-bits are the exponent (base 10).
- The remaining 24-bits are the mantissa.
- Standard positive/negative representations apply for exponent and mantissa.
- A negative exponent gives the number of decimal places to the right of the decimal point.
- There are codes to represent special values.
- Examples:

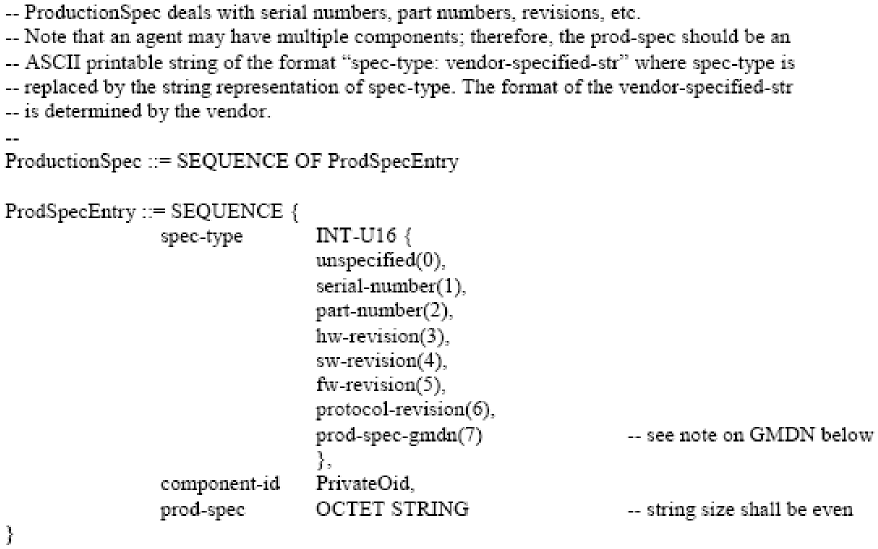
FLOAT	exponent	mantissa	value
0xFE01E240	-2	123456	1234.56
0x0201E240	2	123456	12345600
0x0001E240	0	123456	123456
0xFB000005	-5	5	0.00005
0xFD000005	-3	5	0.005
0xFE00C350	-2	50000	500.00
0xFF001388	-1	5000	500.0
0xFB000000	-5	0	0.00000
0xFD000000	-3	0	0.000
0xFFFE1DC0	-1	-123456	-12345.6
0xFEFFFFFE	-2	-2	-0.02

0x02FFFFFE	2	-2	-200
0x00FFFF38	0	-200	-200
0x007FFFFF			NaN (Not a Number)
0x007FFFFE			+Inf (Positive infinity)
0x00800002			-Inf (Negative infinity)
0x00800000			NRes (Not at this resolution)
0x00800001			Reserved for future use

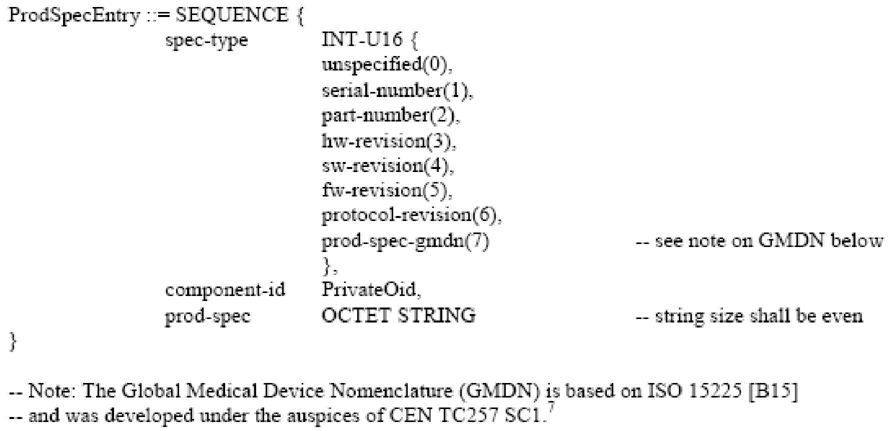
Label	Description	**Release**
HD-HLF-03	The Plug-In SHALL have a real time clock that is synchronized to UTC and SHALL be aware of its local time zone.	1.0
HD-HLF-04	The Plug-In SHALL have a real time clock with a resolution that matches the resolution of any device that it interacts with.	1.0
HD-HLF-05	The Plug-In SHALL be able to obtain the current time from the device if the device reports a current time. (Devices that report a time stamp with their measurements are required to be able to report the device's sense of current time to interoperate with the Plug-In.)	1.0
HD-HLF-06	The Plug-In SHALL be able to map any measurement time stamp reported by the device to an HL7 DTM time stamp with offset from UTC to local time. (An HL7 DTM time stamp is YYYYMMDDHHMMSS.sss+/-HHMM). "DTM" data type that includes the time zone offset, expressed either as ±ZZZZ (HHMM) if the civil time zone offset is known or -0000 if UTC time (e.g. derived from NTP) is known but the actual civil time zone offset is not. **YYYYMMDDHHMMSS[.S[S[S[S]]]]±ZZZZ if civil time zone offset ±ZZZZ is known YYYYMMDDHHMMSS[.S[S[S[S]]]]-0000 if UTC time is known but civil time zone is not.** If the device does not report a time stamp with its measurement, the Plug-In SHALL use the time of reception of the measurement as the measurement time stamp. The Plug-In SHALL provide a Boolean indication of 'true' if the measurement was provided by the Plug-In because the device did not provide a measurement time stamp.	1.0
HD-HLF-07	The Plug-In SHALL correct any *measurement* time stamp by the difference between the current time reported by Device and the current time reported by the Plug-In *unless* the Plug-In knows that the device has a superior synchronization to UTC than the Plug-In does. In other words, if the device does not have superior time synchronization and the current time reported by the device is 20 seconds behind that reported by the Plug-In, the Plug-In adds 20 seconds to any of the measurement time stamps reported by the device. If the device has superior time synchronization, the Plug-In reports the device measurement time stamp unmodified. (Note that PCHA complaint devices have a means of reporting its time synchronization means and state to the Plug-In.) Note: the Plug_in has the responsibility to correct the time. Some devices do not know the time or the time need to be set (i.e. do not have superior synchronization) manually (which may be different from the actual time), so that why the Plug-in need to correct the time	1.0
HD-HLF-08	The Plug-In SHALL have the capability to connect to and interact with PCHA-compliant devices on at least one PCHA-complaint transport. Non-PCHA compliant devices MAY also be supported as long as the following constraints are met: 1. If the proprietary device reports a time stamp with the measurement, the device SHALL have a means of obtaining its current time such that the Plug-In can satisfy guidelines *-HLF-05 to *-HLF-07. 2. If the device stores data a time stamp SHALL be provided with the measurement. Note this requirement also requires the device provide a means to obtain its current time. 3. The device provides sufficient information such that the Plug-In is able to satisfy the remaining requirements.	1.0

Label	Description	Release
HD-HLF-09	The Plug-In SHALL be able to provide the product name of the connected device. If the Plug-In cannot get the product name, it SHALL create a name for the device using an arbitrary algorithm. The algorithm is up to the Plug-In implementation, and this specification does not define any algorithms. Note that the 'algorithm' could be a non-empty user-entry.	1.0

HD-HLF-10	The Plug-In SHALL be able to provide the manufacturer name of the connected device if the Plug-In can get the name. It SHALL be reported as a string (may be empty).
	Note: **MDS attributes**
	
	--
	-- SystemModel contains manufacturer name and manufacturer specific model information.
	-- While model-number field name suggests a number, there is no requirement that the field
	-- contains numeric values. The format of the manufacturer name and model number strings
	-- are decided upon by the agent vendor, but shall be printable ASCII.
	--
	SystemModel ::= SEQUENCE {
	manufacturer OCTET STRING, -- string size shall be even
	model-number OCTET STRING -- string size shall be even
	}
		1.0

HD-HLF-11	The Plug-In SHALL be able to provide the model number of the connected device if the Plug-In can get the model number. It SHALL be reported as a string (may be empty).
	Note: **MDS attributes**
	
	-- ProductionSpec deals with serial numbers, part numbers, revisions, etc.
	-- Note that an agent may have multiple components; therefore, the prod-spec should be an
	-- ASCII printable string of the format "spec-type: vendor-specified-str" where spec-type is
	-- replaced by the string representation of spec-type. The format of the vendor-specified-str
	-- is determined by the vendor.
	--
	ProductionSpec ::= SEQUENCE OF ProdSpecEntry
	
	ProdSpecEntry ::= SEQUENCE {
	spec-type INT-U16 {
	unspecified(0),
	serial-number(1),
	part-number(2),
	hw-revision(3),
	sw-revision(4),
	fw-revision(5),
	protocol-revision(6),
	prod-spec-gmdn(7) -- see note on GMDN below
	},
	component-id PrivateOid,
	prod-spec OCTET STRING -- string size shall be even
	}
		1.0

HD-HLF-12	The Plug-In SHALL be able to provide the firmware revision of the connected device if the Plug-In can get the firmware revision. It SHALL be reported as a string (may be empty).	1.0
HD-HLF-13	The Plug-In SHALL be able to provide the serial number of the connected device if the Plug-In can get the serial number. It SHALL be reported as a string (may be empty).	1.0
HD-HLF-14	The Plug-In SHOULD be able to provide the software revision of the connected device if the Plug-In can get the the software revision. If reported it SHALL be reported as a string.	1.0
HD-HLF-15	The Plug-In SHOULD be able to provide the hardware revision of the connected device if the Plug-In can get the hardware revision. If reported it SHALL be reported as a string.	1.0

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>HD-HLF-16</td><td>The Plug-In SHOULD be able to provide the part number of the connected device if the Plug-In can get the part number. If reported it SHALL be reported as a string.
Note: <b>MDS attributes</b>

<pre>
ProdSpecEntry ::= SEQUENCE {
                 spec-type   INT-U16 {
                                  unspecified(0),
                                  serial-number(1),
                                  part-number(2),
                                  hw-revision(3),
                                  sw-revision(4),
                                  fw-revision(5),
                                  protocol-revision(6),
                                  prod-spec-gmdn(7)   -- see note on GMDN below
                                  },
                 component-id   PrivateOid,
                 prod-spec   OCTET STRING   -- string size shall be even
}

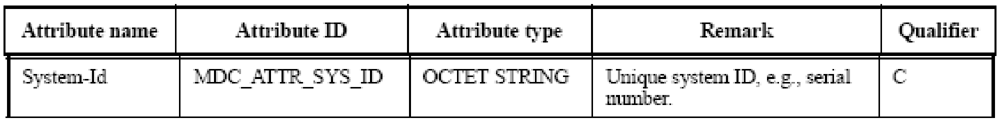
-- Note: The Global Medical Device Nomenclature (GMDN) is based on ISO 15225 [B15]
-- and was developed under the auspices of CEN TC257 SC1.[7]
</pre></td><td>1.0</td></tr>
<tr><td>HD-HLF-17</td><td>The Plug-In SHOULD be able to provide the protocol revision of the connected device if the Plug-In can get the protocol revision. If reported it SHALL be reported as a string.</td><td>1.0</td></tr>
<tr><td>HD-HLF-18</td><td>The Plug-In SHALL be able to provide the 64-bit IEEE system id of the connected device as a 16-character HEX string (without a ‘0x’ prefix). If the device does not report an IEEE system id, the Plug-In SHALL send a string of 16 ‘0’ characters.
<b>Note: VMS object class attributes</b>

<table>
<tr><th>Attribute name</th><th>Attribute ID</th><th>Attribute type</th><th>Remark</th><th>Qualifier</th></tr>
<tr><td>System-Id</td><td>MDC_ATTR_SYS_ID</td><td>OCTET STRING</td><td>Unique system ID, e.g., serial number.</td><td>C</td></tr>
</table></td><td>1.0</td></tr>
<tr><td>HD-HLF-19</td><td>The Plug-In SHALL be able to provide the battery level if the device provides a battery level. This value must be a float number in a range from 0.0 to 1.0.
The value 0.0 represents that the targeted thermometer is completely out of charge. The value 1.0 represents that the targeted thermometer is fully charged.
Even if the targeted thermometer reports this value in percent in a range from 1 to 100, the Plug-In SHALL convert it to a float number in a range from 0.0 to 1.0.
If the Plug-In can't obtain battery level from the targeted thermometer, this value SHALL be -1.0.</td><td>1.0</td></tr>
</table>

Table 1: High-Level Functional Requirements

6.3 Thermometer Specific Functional Requirements

The following requirements outline the thermometer specific set of options that Thermometer Plug-Ins implement. The Thermometer Plug-In technical specifications will address the necessary functions for support of these options. This device typically would be what one calls a 1 – N shot device where N is less than 25. However, if the device stores data persistently, the number of measurements could be very large.

Editor's note:

This section should specify all the detailed and necessary requirements that are specific to the profile, so that the architecture and technical specifications can be developed.

Thermometer devices supported by this plug in specification are expected to be able to report the body temperature. The description of the measurement reported by the plug in follows the IEEE 11073 10408 Thermometer specialization specification but that does not mean the device itself must follow that specification. However the device must provide to the plug in the necessary information such that the plug can fulfil its reporting requirements as specified in this document.

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>T-HSF-00</td><td>The Plug-In SHALL provide values as both strings (human consumption) and MDC codes for detailed understanding and machine processing).
Note: For each nomenclature term, ISO/IEEE 11073-10101 [B12] defines a systematic name that explains the term, a unique code value, and a reference identifier (ID). The reference ID has the form MDC_XXX_YYY (with MDC referring to “medical device communication”). Throughout this standard, nomenclature terms and nomenclature codes are referenced by the reference ID.</td><td>1.0</td></tr>
<tr><td>T-HSF-01</td><td>The Temperature Plug-In SHALL use the Body temperature numeric object attributes to report the values for subsequent requirements i.e. T-HSF-01.1...etc as stated in these guidelines for the equivalent of the Body Temperature object as defined in IEEE 11073 10408 Table 5
Note: <b>Body temperature numeric object attributes</b>
<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th><th colspan="2">Standard configuration</th></tr>
<tr><th>Value</th><th>Qual</th><th>Value</th><th>Qual</th></tr>
<tr><td>Type</td><td>{MDC_PART_SCADA,
MDC_TEMP_zzz}.</td><td>M</td><td>{MDC_PART_SCADA,
MDC_TEMP_BODY}.</td><td>M</td></tr>
<tr><td>Metric-Id</td><td>See IEEE Std 11073-20601.</td><td><b>C</b></td><td>Attribute not initially present. If present follow IEEE Std 11073-20601.</td><td><b>NR</b></td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601.</td><td>C</td><td>Attribute not initially present. If present follow IEEE Std 11073-20601.</td><td>C</td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_DEGC or
MDC_DIM_FAHR.</td><td>M</td><td>MDC_DIM_DEGC.</td><td>M</td></tr>
</table>
</td><td>1.0</td></tr>
<tr><td>T-HSF-01.1</td><td>The Temperature Plug-In SHALL report the value of the TYPE* attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition: code) *If the Metric Id is used the Temperature Plug-In SHALL replace the code value with this value and if the Metric-Id-partition is present the partition value SHALL be replaced with this value.
Note: TYPE and Metric Id identify the type of device but Metric Id gives more information.
Example: String: “Oral body temperature”
Code: “188424</td><td>1.0</td></tr>
</table>

Label	Description	Release
T-HSF-01.2	The Thermometer Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string: 1- Float that represents the temperature measured by the targeted thermometer and represented in a float format. This is for general use of application programming. 2- String that represents the temperature measured by the targeted thermometer and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT. Example: String: “37.2” MDER FLOAT:“FFFFC8E”	1.0
T-HSF-01.3	The Temperature Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). Example: String: “deg C” Code: “268192”	1.0
T-HSF-01.4	The Temperature Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07. Example: “20150504135813.22-0400”	1.0

Table 2: Thermometer Specific Functional Requirements

Editor Note: Thermometer

Type as String: “Oral body temperature” Type as Code: “188424”	
Type as String: “37.2” MDER FLOAT:“FFFFC8E”	Units as string: “deg C” Units as code: “268192”
Timestamp: “20150504135813.22-0400”	

6.4 Pulse Oximeter Specific Functional Requirements

The following requirements outline the pulse oximeter specific set of options that Pulse Oximeter Plug-Ins may implement. The Pulse Oximeter Plug-In technical specifications will address the necessary functions for support of these options. Pulse Oximeters may stream data or send spot data episodically similar to a Blood Pressure cuff or do both.

Editor's note:

This section should speficy all the detailed and necessary requirements that are specific to the profile, so that the architecture and technical specifications can be developed.

Pulse oximeter devices supported by this plug in specification are expected to be able to report the oxygen saturation and pulse rate. The description of the measurement reported by the plug in follows the IEEE 11073 10404 Pulse Oximeter specialization specification but that does not mean the device itself must follow that specification. However the device must provide to the plug in the necessary information such that the plug can fulfil its reporting requirements as specified in this document.

Pulse oximeters report measurements in several different manners. Data is often streamed at regular intervals (say once per second) or reported over various ‘sample’ times. The latter are referred to as spot modalities, and they can be fast, slow, or just ‘spot’. The spot measurements represent a more ‘robust’ estimate of the actual value. IEEE supports several types of spot modality measurements. Spot measurements, being episodic, typically have time stamps whereas streaming measurements tend not to have time stamps.

The Pulse Oximeter Plug-In will report at least one type of oxygen saturation and pulse rate. It may be further described as modality spot, modality fast spot, or modality slow spot.

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>PO-HSF-00</td><td>The Plug-In SHALL provide values as both strings (human consumption) and MDC codes for detailed understanding and machine processing).</td><td>1.0</td></tr>
<tr><td>PO-HSF-01</td><td>The Pulse Oximeter Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Oxygen Saturation object as defined in IEEE 11073 10404 Table 5.
Note: <b>SpO2 numeric object attributes</b>
<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th><th colspan="2">Standard configuration</th></tr>
<tr><th>Value</th><th>Qual</th><th>Value</th><th>Qual</th></tr>
<tr><td>Type</td><td>{MDC_PART_SCADA,
MDC_PULS_OXIM_SAT_O2}</td><td>M</td><td>{MDC_PART_SCADA,
MDC_PULS_OXIM_SAT_O2}</td><td>M</td></tr>
<tr><td>Supplemental-Types</td><td>The Supplemental-Types attribute is used to distinguish the modality of a particular SpO2 measurement. In order to express the fast-response SpO2 measurement, slow-response SpO2 measurement, If there is no desire to distinguish a modality, the Supplemental-Types attribute shall not be used.</td><td><b>C</b></td><td>See IEEE Std 11073-20601-2008 and following text.</td><td><b>NR</b></td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601-2008.</td><td>C</td><td>Attribute not initially present. If present, follow IEEE Std 11073-20601-2008.</td><td>C</td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_PERCENT</td><td>M</td><td>MDC_DIM_PERCENT</td><td>M</td></tr>
</table></td><td>1.0</td></tr>
<tr><td>PO-HSF-01.1</td><td>The Pulse Oximeter Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition: code). <i>[The TYPE attribute value for the Oxygen Saturation is fixed for all Pulse Oximeter devices]</i>
Example: String: “Oxygen Saturation”
Code: “150456”
[MDC_PULS_OXIM_SAT_O2]</td><td>1.0</td></tr>
</table>

Label	Description	Release
PO-HSF-01.1.1	The Pulse Oximeter Plug-In SHALL report the value of the Supplemental types attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code with the 16-bit code partition:code) if a supplemental types exists. Example: String: “Spot (average) measurement” Code: “150588” [MDC_MODALITY_SPOT]	1.0
PO-HSF-01.2	The Pulse Oximeter Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string: 1- Float that represents the Pulse Oximeter report data measured by the targeted Pulse Oximeter and represented in a float format. This is for general use of application programming. 2- String that represents the Pulse Oximeter report data measured by the targeted Pulse Oximeter and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT. Example: String: “98” MDER FLOAT: “00000062’	1.0
PO-HSF-01.3	The Pulse Oximeter Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). *[The Unit Code attribute value for the Oxygen Saturation is fixed for all Pulse Oximeter devices]* Example: String: “%” Code: “262688” [MDC_DIM_PERCENT]	1.0
PO-HSF-01.4	The Pulse Oximeter Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07. Example: 20150506162223.50-0400	

Table 3: Pulse Oximeter Specific Functional Requirements

Editor Note: Oxygen Saturation

Type as String: “Oxygen Saturation” Type as Code: “150456”	
Type as String: “Spot (average) measurement” Code: “150588	
Type as String: “98” MDER FLOAT: “00000062’	Units as string: “%” Units as code: “262688”
Timestamp: “20150504135813.22-0400”	

Editor Note: Pulse Rate

Type as String: “Pulse Rate” Type as Code: “149530”	
Type as String: “Spot (average) measurement” Code: “150588	
Type as String: “42” MDER FLOAT: “0000002A”	Units as string: “beats per min” Units as code: “264864”
Timestamp: “20150504135813.22-0400”	

6.5 Weight Scale / Body Composition Analyzer Functional Requirements

The following requirements outline the weight scale and body composition analyzer specific set of options that Weight Scale / Body Composition Analyzer Plug-Ins implement. The Weight Scale / Body Composition Analyzer Plug-In technical specifications will address the necessary functions for support of these options. This device typically would be what one calls a 1 – N shot device where N is less than 25. However, if the device stores data persistently, the number of measurements could be very large.

Editor's note:

This section should speficy all the detailed and necessary requirements that are specific to the profile, so that the architecture and technical specifications can be developed.

Weight Scale and Body Composition Analyzer (BCA) plug ins are specified together as a BCA has all the required and optional measurements of a Weight Scale device in addition to additional required and optional measurements.

Weight Scale devices supported by the Weight Scale plug in specification are expected to be able to report the body mass and optionally the body length (height) and body mass index (BMI). The description of the measurements reported by the Weight Scale plug in follows the IEEE 11073 10415 Weight Scale specialization specification but that does not mean the device itself must follow that specification. However the device must provide to the plug in the necessary information such that the plug can fulfil its reporting requirements as specified in this document.

Body Composition Analyzers (BCA) report body fat, body mass, body length, and may support several other related measurements such as muscle mass, body water, fat free mass, soft lean mass, and BMI as specified in IEEE 11073 10420. The Bluetooth Low Energy BCA service specification has additional measurements not defined in IEEE though MDC codes have been defined for these Bluetooth Low Energy measurements and it is expected that they will be added to the IEEE specification in future revisions of the IEEE specification.

Given the fact that a BCA is essentially a weight scale with additional measurements, the BCA Plug-In will support all the Weight Scale Plug-In guidelines in addition to those guidelines specific to the BCA. The only inconsistency is that IEEE BCAs require a body length, and in IEEE weight scales it is optional. Thus if the BCA Plug-In does not receive a body length measurement, then it will not report such a measurement.

The example column shows what the Plug-In will send to the upper levels. There will be a string and a Code or MDER FLOAT if the latter apply. The string representation of the MDC reference code in square brackets is NOT passed up but shown for reference.

Label	Description	Release
WS-HSF-00	The Plug-In SHALL provide values as both strings (human consumption) and MDC codes for detailed understanding and machine processing).	1.0

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>WS-HSF-01</td><td>The Weight Scale Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Body Mass object as defined in IEEE 11073 10415 Table 5.
Note: <b>weight scale</b>
<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th><th colspan="2">Standard configuration</th></tr>
<tr><th>Value</th><th>Qual</th><th>Value</th><th>Qual</th></tr>
<tr><td>Type</td><td>MDC_PART_SCADA |
MDC_MASS_BODY_ACTUAL.</td><td>M</td><td>MDC_PART_SCADA |
MDC_MASS_BODY_ACTUAL.</td><td>M</td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601.</td><td><b>C</b></td><td>Attribute not initially present. If present
follow IEEE Std 11073-20601.</td><td><b>C</b></td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_KILO_G or
MDC_DIM_LB.</td><td>M</td><td>MDC_DIM_KILO_G</td><td>M</td></tr>
<tr><td></td><td></td><td>M</td><td></td><td>M</td></tr>
</table>
</td><td>1.0</td></tr>
<tr><td>WS-HSF-01.1</td><td>The Weight Scale Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). <i>[The TYPE attribute value for the Body Mass is fixed for all Weight Scale devices]</i>
Example: String: “Body Mass”
Code: “188740”
[MDC_MASS_BODY_ACTUAL]</td><td>1.0</td></tr>
<tr><td>WS-HSF-01.2</td><td>The Weight Scale Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string:
Float that represents the Weight Scale report data measured by the targeted Weight Scale and represented in a float format. This is for general use of application programming.
String that represents the Weight Scale report data measured by the targeted Weight Scale and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT.
Example: String: “160.4”
MDER FLOAT: “FF00644”</td><td>1.0</td></tr>
<tr><td>WS-HSF-01.3</td><td>The Weight Scale Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code).
Example: String: “lbs”
Code: “263904”
[MDC_DIM_LB]</td><td>1.0</td></tr>
</table>

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>WS-HSF-01.4</td><td>The Weight Scale Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07.
Example: “20150506135813.22-0400”</td><td>1.0</td></tr>
<tr><td>WS-HSF-02</td><td>The Weight Scale Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Body Length object as defined in IEEE 11073 10415 Table 6 <b><i>if</i></b> the device reports a body length.
Note: <b>Body height numeric object attributes</b>
<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th>Value</th><th>Qual</th></tr>
<tr><td>Type</td><td>MDC_PART_SCADA |
MDC_LEN_BODY_ACTUAL.</td><td>M</td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601.</td><td><b>C</b></td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_CENTI_M, or
MDC_DIM_INCH.</td><td>M</td></tr>
</table>
</td><td>1.0</td></tr>
<tr><td>WS-HSF-02.1</td><td>If the device reports a body length, the Weight Scale Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). <i>[The TYPE attribute value for the Body Length is fixed for all Weight Scale devices]</i>
Example: String: “Body Length”
Code: “188744”
[MDC_LEN_BODY_ACTUAL]</td><td>1.0</td></tr>
</table>

Label	Description	Release
WS-HSF-02.2	If the device reports a body length, the Weight Scale Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string: Float that represents the body length report data measured by the targeted Weight Scale and represented in a float format. This is for general use of application programming. String that represents the body length report data measured by the targeted Weight Scale and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT. Example: String: “68.5” MDER FLOAT: “FF002AD”	1.0
WS-HSF-02.3	If the device reports a body length, the Weight Scale Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). Example: String: “inches” Code: “263520” [MDC_DIM_INCH]	1.0
WS-HSF-02.4	If the device reports a body length, the Weight Scale Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07. Example: “20150506135813.22-0400”	1.0

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>WS-HSF-03</td><td>The Weight Scale Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Body Mass Index (BMI) object as defined in IEEE 11073 10415 Table 7 <b><i>if</i></b> the device reports a BMI.
Note: <b>Body mass index numeric object attributes</b>
<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th>Value</th><th>Qual</th></tr>
<tr><td>Type</td><td>MDC_PART_SCADA |
MDC_RATIO_MASS_BODY_LEN_SQ.</td><td>M</td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601.</td><td><b>C</b></td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_KG_PER_M_SQ.</td><td>M</td></tr>
</table>
</td><td>1.0</td></tr>
<tr><td>WS-HSF-03.1</td><td>If the device reports a BMI, the Weight Scale Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). <i>[The TYPE attribute value for the Body Mass Index is fixed for all Weight Scale devices]</i>
Example: String: “BMI”
Code: “188752”
[MDC_RATIO_MASS_BODY_LEN_SQ]</td><td>1.0</td></tr>
<tr><td>WS-HSF-03.2</td><td>If the device reports a BMI, the Weight Scale Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string:
Float that represents the BMI report data measured by the targeted Weight Scale and represented in a float format. This is for general use of application programming.
String that represents the BMI report data measured by the targeted Weight Scale and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT.
Example: String: “24.10”
MDER FLOAT: “FE00096A”</td><td>1.0</td></tr>
</table>

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>WS-HSF-03.3</td><td>If the device reports a BMI, the Weight Scale Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code).
Example:
String: “kg/m^2”
Code: “264096”
[MDC_DIM_KG_PER_M_SQ]</td><td>1.0</td></tr>
<tr><td>WS-HSF-03.4</td><td>If the device reports a BMI, the Weight Scale Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07.
Example: “20150506135813.22-0400”</td><td>1.0</td></tr>
<tr><td>BCA-HSF-01</td><td>The BCA Plug in SHALL support the Weight Scale Plug-In guidelines as stated in this specification with the note that IEEE BCA devices are <i>expected</i> to support the body length though the Bluetooth Low Energy BCA service has the body length as optional. Consequently the BCA Plug-in SHALL report the body length measurement if the device sends it.</td><td>1.0</td></tr>
<tr><td>BCA-HSF-02</td><td>The BCA Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Body Fat object as defined in IEEE 11073 10420 Table 5.
Note: <b>BCA index numeric object attributes</b>
<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th>Value</th><th>Qual</th></tr>
<tr><td>Type</td><td>MDC_BODY_FAT</td><td>M</td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601.</td><td><b>C</b></td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_PERCENT</td><td>M</td></tr>
</table></td><td>1.0</td></tr>
<tr><td>BCA-HSF-02.1</td><td>The BCA Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). <i>[The TYPE attribute value for the Body Fat is fixed for all BCA devices]</i>
Example: String: “Body Fat”
Code: “188748”
[MDC_BODY_FAT]</td><td>1.0</td></tr>
</table>

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>BCA-HSF-02.2</td><td>The BCA Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string:
Float that represents the BCA report data measured by the targeted BCA and represented in a float format. This is for general use of application programming.
String that represents the BCA report data measured by the targeted BCA and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT.
Example: String: “13.5”
MDER FLOAT: “FF000087”</td><td>1.0</td></tr>
<tr><td>BCA-HSF-02.3</td><td>The BCA Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code).
Example: String: “%”
Code: “262688”
[MDC_DIM_PERCENT]</td><td>1.0</td></tr>
<tr><td>BCA-HSF-02.4</td><td>The BCA Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07.
Example: “20150506135813.22-0400”</td><td>1.0</td></tr>
<tr><td>BCA-HSF-03</td><td>The BCA Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Fat Free Mass object as defined in IEEE 11073 10420 Table 8 if the device reports the measurement.
Note: <b>Fat Free index numeric object attributes</b>
<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th>Value</th><th>Qual</th></tr>
<tr><td>Type</td><td>MDC_MASS_BODY_FAT_FREE</td><td>M</td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601.</td><td><b>C</b></td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_LB</td><td>M</td></tr>
</table></td><td>1.0</td></tr>
</table>

Label	Description	Release
BCA-HSF-03.1	If the device reports a fat free mass measurement, the BCA Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). *[The TYPE attribute value for the Fat Free Mass is fixed for all BCA devices]* Example: String: “Fat Free Mass” Code: “188756” [MDC_MASS_BODY_FAT_FREE]	1.0
BCA-HSF-03.2	If the device reports a fat free mass measurement, the BCA Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string: Float that represents the fat free mass measurement report data measured by the targeted BCA and represented in a float format. This is for general use of application programming. String that represents the fat free mass measurement report data measured by the targeted BCA and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT. Example: String: “138.8” MDER FLOAT: “FF00056C”	1.0
BCA-HSF-03.3	If the device reports a fat free mass measurement, the BCA Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). Example: String: “lbs” Code: “263904” [[MDC_DIM_LB]	1.0
BCA-HSF-03.4	If the device reports a fat free mass measurement, the BCA Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07. Example: “20150506135813.22-0400”	1.0

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>BCA-HSF-04</td><td>The BCA Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Soft Lean Mass object as defined in IEEE 11073 10420 Table 9 if the device reports the measurement.
Note: **Soft Lean Mass index numeric object attributes**
<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th>Value</th><th>Qual</th></tr>
<tr><td>Type</td><td>[MDC_MASS_BODY_SOFT_LEAN]</td><td>M</td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601.</td><td>**C**</td></tr>
<tr><td>Unit-Code</td><td>[MDC_DIM_KILO_G]</td><td>M</td></tr>
</table></td><td>1.0</td></tr>
<tr><td>BCA-HSF-04.1</td><td>If the device reports a soft lean mass measurement, the BCA Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). *[The TYPE attribute value for the Soft Lean Mass is fixed for all BCA devices]*
Example: String: “Soft Lean Mass”
Code: “188760”
[MDC_MASS_BODY_SOFT_LEAN]</td><td>1.0</td></tr>
<tr><td>BCA-HSF-04.2</td><td>If the device reports a soft lean mass measurement, the BCA Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string:
Float that represents a soft lean mass measurement report data measured by the targeted BCA and represented in a float format. This is for general use of application programming.
String that represents a soft lean mass measurement report data measured by the targeted BCA and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT
Example: String: “58.8”
MDER FLOAT: “FF00024C”</td><td>1.0</td></tr>
<tr><td>BCA-HSF-04.3</td><td>If the device reports a soft lean mass measurement, the BCA Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code).
Example: String: “kg”
Code: “263875”
[MDC_DIM_KILO_G]</td><td>1.0</td></tr>
</table>

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>BCA-HSF-04.4</td><td>If the device reports a soft lean mass measurement, the BCA Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07.
Example: “20150506135813.22-0400”</td><td>1.0</td></tr>
<tr><td>BCA-HSF-05</td><td>The BCA Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Body Water object as defined in IEEE 11073 10420 Table 10 if the device reports the measurement.
Note: <b>Body Water object index numeric object attributes</b>
<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th>Value</th><th>Qual</th></tr>
<tr><td>Type</td><td>[MDC_BODY_WATER]</td><td>M</td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601.</td><td><b>C</b></td></tr>
<tr><td>Unit-Code</td><td>[MDC_DIM_PERCENT]</td><td>M</td></tr>
</table></td><td>1.0</td></tr>
<tr><td>BCA-HSF-05.1</td><td>If the device reports a body water measurement, the BCA Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). <i>[The TYPE attribute value for the Body Water is fixed for all BCA devices]</i>
Example: String: “Body water”
Code: “188760”
[MDC_BODY_WATER]</td><td>1.0</td></tr>
<tr><td>BCA-HSF-05.2</td><td>If the device reports a body water measurement, the BCA Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string:
Float that represents a body water measurement report data measured by the targeted BCA and represented in a float format. This is for general use of application programming.
String that represents a body water measurement report data measured by the targeted BCA and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT
Example: String: “64”
MDER FLOAT: “00000040”</td><td>1.0</td></tr>
</table>

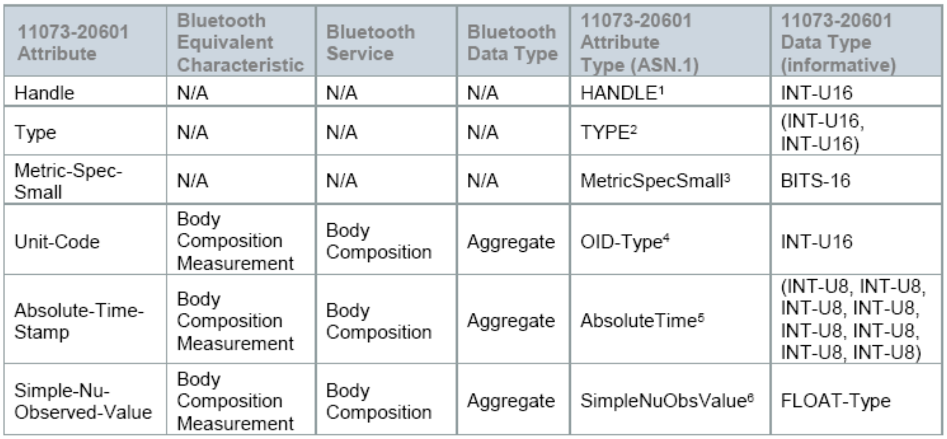
<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>BCA-HSF-05.3</td><td>If the device reports a body water measurement, the BCA Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code).
Example: String: “%”
Code: “262688”
[MDC_DIM_PERCENT]</td><td>1.0</td></tr>
<tr><td>BCA-HSF-05.4</td><td>If the device reports a body water measurement, the BCA Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07.
Example: “20150506135813.22-0400”</td><td>1.0</td></tr>
<tr><td>BCA-HSF-06</td><td>The BCA Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Muscle Mass as defined in Bluetooth Low Energy BCA Service and the Personal Health Devices Transcoding White Paper Table 61 if the device reports the measurement. The IEEE BCA specialization does not currently support this measurement though an MDC code (188776; 2:E168) is provided for it.
Note: <b>Muscle Mass object index numeric object attributes</b>
<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th>Value</th><th>Qua l</th></tr>
<tr><td>Type</td><td>[MDC_BODY_MASS]</td><td>M</td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601.</td><td><b>C</b></td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_KILO_G</td><td>M</td></tr>
</table></td><td>1.0</td></tr>
<tr><td>BCA-HSF-06.1</td><td>If the device reports a muscle mass measurement, the BCA Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). <i>[The TYPE attribute value for the Muscle Mass is fixed for all BCA devices]</i>
Example: String: “Muscle Mass”
Code: “188776”</td><td>1.0</td></tr>
</table>

Label	Description	Release
BCA-HSF-06.2	If the device reports a muscle mass measurement, the BCA Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string: Float that represents a muscle mass measurement report data measured by the targeted BCA and represented in a float format. This is for general use of application programming. String that represents a muscle mass measurement report data measured by the targeted BCA and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT Example: String: “43” MDER FLOAT: “0000002B	1.0
BCA-HSF-06.3	If the device reports a muscle mass measurement, the BCA Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). Example: String: “kg” Code: “263875” [MDC_DIM_KILO_G]	1.0
BCA-HSF-06.4	If the device reports a muscle mass measurement, the BCA Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07. Example: “20150506135813.22-0400”	1.0

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>BCA-HSF-07</td><td>
The BCA Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Muscle percentage as defined in Bluetooth Low Energy BCA Service and the Personal Health Devices Transcoding White Paper Table 60 if the device reports the measurement. The IEEE BCA specialization does not currently support this measurement though an MDC code (188772; 2:E164) is provided for it.

Note: **Muscle Percentage object index numeric object attributes**

<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th>Value</th><th>Qual</th></tr>
<tr><td>Type</td><td>[MDC_BODY_ MUSCLE_PERCENTAGE]</td><td>M</td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601.</td><td><b>C</b></td></tr>
<tr><td>Unit-Code</td><td>[MDC_DIM_PERCENT]</td><td>M</td></tr>
</table>



<table>
<tr><th>11073-20601 Attribute</th><th>Bluetooth Equivalent Characteristic</th><th>Bluetooth Service</th><th>Bluetooth Data Type</th><th>11073-20601 Attribute Type (ASN.1)</th><th>11073-20601 Data Type (informative)</th></tr>
<tr><td>Handle</td><td>N/A</td><td>N/A</td><td>N/A</td><td>HANDLE[1]</td><td>INT-U16</td></tr>
<tr><td>Type</td><td>N/A</td><td>N/A</td><td>N/A</td><td>TYPE[2]</td><td>(INT-U16, INT-U16)</td></tr>
<tr><td>Metric-Spec-Small</td><td>N/A</td><td>N/A</td><td>N/A</td><td>MetricSpecSmall[3]</td><td>BITS-16</td></tr>
<tr><td>Unit-Code</td><td>Body Composition Measurement</td><td>Body Composition</td><td>Aggregate</td><td>OID-Type[4]</td><td>INT-U16</td></tr>
<tr><td>Absolute-Time-Stamp</td><td>Body Composition Measurement</td><td>Body Composition</td><td>Aggregate</td><td>AbsoluteTime[5]</td><td>(INT-U8, INT-U8, INT-U8, INT-U8, INT-U8, INT-U8, INT-U8, INT-U8)</td></tr>
<tr><td>Simple-Nu-Observed-Value</td><td>Body Composition Measurement</td><td>Body Composition</td><td>Aggregate</td><td>SimpleNuObsValue[6]</td><td>FLOAT-Type</td></tr>
</table>

Table 60: Numeric Class: Muscle Percentage
</td><td>1.0</td></tr>
<tr><td>BCA-HSF-07.1</td><td>
If the device reports a muscle percentage measurement, the BCA Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). *[The TYPE attribute value for the Muscle percentage is fixed for all BCA devices]*

Example: String: “Muscle Percentage”

Code: “188772”
</td><td>1.0</td></tr>
</table>

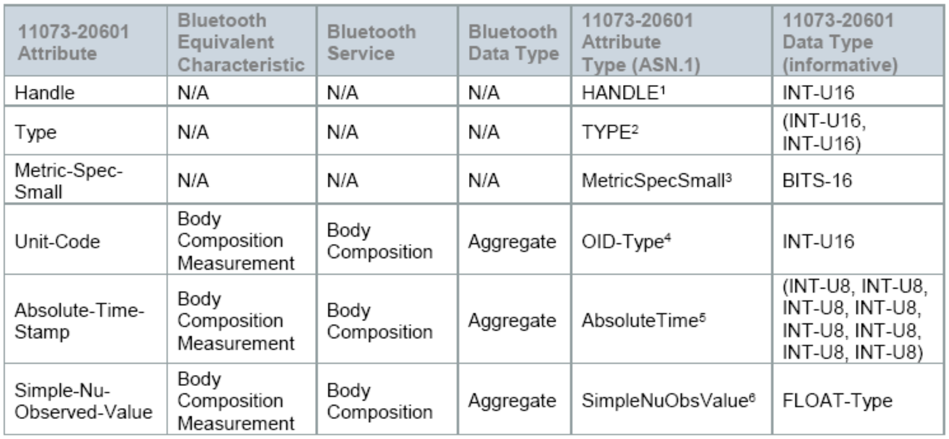
Label	Description	Release
BCA-HSF-07.2	If the device reports a muscle percentage measurement, the BCA Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string: Float that represents a muscle percentage measurement report data measured by the targeted BCA and represented in a float format. This is for general use of application programming. String that represents a muscle percentage measurement report data measured by the targeted BCA and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT Example: String: “59” MDER FLOAT: “0000003B”	1.0
BCA-HSF-07.3	If the device reports a muscle percentage measurement, the BCA Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). *[The Unit code attribute value for the Muscle percentage is fixed for all BCA devices]* Example: String: “%” Code: “262688” [MDC_DIM_PERCENT]	1.0
BCA-HSF-07.4	If the device reports a muscle percentage measurement, the BCA Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07. Example: “20150506135813.22-0400”	1.0

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>BCA-HSF-08</td><td>

The BCA Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Basal Metabolism as defined in Bluetooth Low Energy BCA Service and the Personal Health Devices Transcoding White Paper Table 59 if the device reports the measurement. The IEEE BCA specialization does not currently support this measurement though an MDC code (188768; 2:E160) is provided for it.

Note: **Basal Metabolism bject index numeric object attributes**

<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th>Value</th><th>Qual</th></tr>
<tr><td>Type</td><td>[MDC_BODY_ Basel_Metabolism]</td><td>M</td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601.</td><td><b>C</b></td></tr>
<tr><td>Unit-Code</td><td>[MDC_DIM_PERCENT]</td><td>M</td></tr>
</table>



<table>
<tr><th>11073-20601 Attribute</th><th>Bluetooth Equivalent Characteristic</th><th>Bluetooth Service</th><th>Bluetooth Data Type</th><th>11073-20601 Attribute Type (ASN.1)</th><th>11073-20601 Data Type (informative)</th></tr>
<tr><td>Handle</td><td>N/A</td><td>N/A</td><td>N/A</td><td>HANDLE[1]</td><td>INT-U16</td></tr>
<tr><td>Type</td><td>N/A</td><td>N/A</td><td>N/A</td><td>TYPE[2]</td><td>(INT-U16, INT-U16)</td></tr>
<tr><td>Metric-Spec-Small</td><td>N/A</td><td>N/A</td><td>N/A</td><td>MetricSpecSmall[3]</td><td>BITS-16</td></tr>
<tr><td>Unit-Code</td><td>Body Composition Measurement</td><td>Body Composition</td><td>Aggregate</td><td>OID-Type[4]</td><td>INT-U16</td></tr>
<tr><td>Absolute-Time-Stamp</td><td>Body Composition Measurement</td><td>Body Composition</td><td>Aggregate</td><td>AbsoluteTime[5]</td><td>(INT-U8, INT-U8, INT-U8, INT-U8, INT-U8, INT-U8, INT-U8, INT-U8)</td></tr>
<tr><td>Simple-Nu-Observed-Value</td><td>Body Composition Measurement</td><td>Body Composition</td><td>Aggregate</td><td>SimpleNuObsValue[6]</td><td>FLOAT-Type</td></tr>
</table>

Table 59: Numeric Class: Basal Metabolism

</td><td>1.0</td></tr>
<tr><td>BCA-HSF-08.1</td><td>

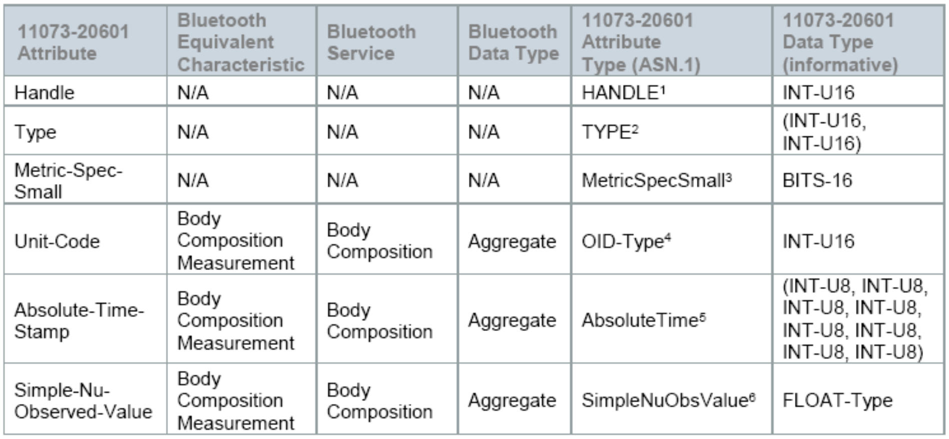
If the device reports a basal metabolism measurement, the BCA Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). *[The TYPE attribute value for the basal metabolism is fixed for all BCA devices]*

Example: String: “Basal Metabolism”

Code: “188768”

</td><td>1.0</td></tr>
</table>

Label	Description	Release
BCA-HSF-08.2	If the device reports a basal metabolism measurement, the BCA Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string: Float that represents a basal metabolism measurement report data measured by the targeted BCA and represented in a float format. This is for general use of application programming. String that represents a basal metabolism measurement report data measured by the targeted BCA and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT Example: String: “1214000” MDER FLOAT: “030004BE”	1.0
BCA-HSF-08.3	If the device reports a basal metabolism measurement, the BCA Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). *[The Unit code attribute value for the basal metabolism is fixed for all BCA devices]* Example: String: “joules” Code: “266112” [MDC_DIM_JOULES]	1.0
BCA-HSF-08.4	If the device reports a basal metabolism measurement, the BCA Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07. Example: “20150506135813.22-0400”	1.0

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>BCA-HSF-09</td><td>
<p>The BCA Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Impedance as defined in Bluetooth Low Energy BCA Service and the Personal Health Devices Transcoding White Paper Table 58 if the device reports the measurement. The IEEE BCA specialization does not currently support this measurement though an MDC code (188780; 2:E16C) is provided for it.</p>
<p>Note: <b>Impedance object index numeric object attributes</b></p>
<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th>Value</th><th>Qual</th></tr>
<tr><td>Type</td><td>[MDC_BODY_ IMPEDANCE]</td><td>M</td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601.</td><td><b>C</b></td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_OHMS</td><td>M</td></tr>
</table>

<table>
<tr><th>11073-20601 Attribute</th><th>Bluetooth Equivalent Characteristic</th><th>Bluetooth Service</th><th>Bluetooth Data Type</th><th>11073-20601 Attribute Type (ASN.1)</th><th>11073-20601 Data Type (informative)</th></tr>
<tr><td>Handle</td><td>N/A</td><td>N/A</td><td>N/A</td><td>HANDLE[1]</td><td>INT-U16</td></tr>
<tr><td>Type</td><td>N/A</td><td>N/A</td><td>N/A</td><td>TYPE[2]</td><td>(INT-U16, INT-U16)</td></tr>
<tr><td>Metric-Spec-Small</td><td>N/A</td><td>N/A</td><td>N/A</td><td>MetricSpecSmall[3]</td><td>BITS-16</td></tr>
<tr><td>Unit-Code</td><td>Body Composition Measurement</td><td>Body Composition</td><td>Aggregate</td><td>OID-Type[4]</td><td>INT-U16</td></tr>
<tr><td>Absolute-Time-Stamp</td><td>Body Composition Measurement</td><td>Body Composition</td><td>Aggregate</td><td>AbsoluteTime[5]</td><td>(INT-U8, INT-U8, INT-U8, INT-U8, INT-U8, INT-U8, INT-U8, INT-U8)</td></tr>
<tr><td>Simple-Nu-Observed-Value</td><td>Body Composition Measurement</td><td>Body Composition</td><td>Aggregate</td><td>SimpleNuObsValue[6]</td><td>FLOAT-Type</td></tr>
</table>
<p><i>Table 58: Numeric Class: Impedance</i></p>
</td><td>1.0</td></tr>
<tr><td>BCA-HSF-08.1</td><td>
<p>If the device reports a basal metabolism measurement, the BCA Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). <i>[The TYPE attribute value for the impedance is fixed for all BCA devices]</i></p>
<p>Example: String: “Impedance”</p>
<p>Code: “188780”</p>
</td><td>1.0</td></tr>
</table>

Label	Description	Release
BCA-HSF-09.2	If the device reports an impedance measurement , the BCA Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string: Float that represents an impedance measurement report data measured by the targeted BCA and represented in a float format. This is for general use of application programming. String that represents an impedance measurement report data measured by the targeted BCA and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT Example: String: “4567.8” MDER FLOAT: “FF00B26E”	1.0
BCA-HSF-09.3	If the device reports an impedance measurement, the BCA Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). *[The Unit code attribute value for the impedance is fixed for all BCA devices]* Example: String: “joules” Code: “266432” [MDC_DIM_OHMS]	1.0
BCA-HSF-09.4	If the device reports an impedance measurement, the BCA Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07. Example: “20150506135813.22-0400”	1.0

Editor Note: Body Mass

Type as String: “Body Mass” Type as Code: “188740”	
Type as String: “160.4” MDER FLOAT: “FF00644”	Units as string: “lbs” Units as code “Code: “263904”
Timestamp: “20150504135813.22-0400”	

Editor Note: Body Length

Type as String: “Body Length” Type as Code: “188744”	
Type as String: “68.5” MDER FLOAT: “FF002AD”	Units as string: “inches” Units as code: “263520”
Timestamp: “20150504135813.22-0400”	

Editor Note: Body Mass Index (BMI)

Type as String: “BMI”	

Type as Code: "188752"	
Type as String: "24.10" MDER FLOAT: "FE00096A"	Units as string: "kg/m2" Units as code: "264096"
Timestamp: "20150504135813.22-0400"	

Editor Note: Body Fat

Type as String: "Body Fat" Type as Code: "188748"	
Type as String: "13.5" MDER FLOAT: "FF000087"	Units as string: "%" Units as code: "262688"
Timestamp: "20150504135813.22-0400"	

Editor Note: Fat Free Mass

Type as String: "Fat Free Mass" Type as Code: "188756"	
Type as String: "138.8" MDER FLOAT: "FF00056C"	Units as string: "lbs" Units as code: "263904"
Timestamp: "20150504135813.22-0400"	

Editor Note: Soft Lean Mass

Type as String: "Soft Lean Mass" Type as Code: "188760"	
Type as String: "58.8" MDER FLOAT: "FF00024C"	Units as string: "kg" Units as code: "263875"
Timestamp: "20150504135813.22-0400"	

Editor Note: Body Water

Type as String: "Body water" Type as Code: "188760"	
Type as String: "64" MDER FLOAT: "00000040"	Units as string: "%" Units as code: "262688"
Timestamp: "20150504135813.22-0400"	

Editor Note: Muscle Mass

Type as String: "Muscle Mass" Type as Code: "188776"	
Type as String: "43" MDER FLOAT: "0000002B"	Units as string: "kg" Units as code: "263875"
Timestamp: "20150504135813.22-0400"	

Editor Note: Muscle percentage

Type as String: "Muscle Percentage" Type as Code: "188772"	
Type as String: "59" MDER FLOAT: "0000003B"	Units as string: "%" Units as code: "262688"

Timestamp: “20150504135813.22-0400”	

Editor Note: Basal Metabolism

Type as String: “Basal Metabolism” Type as Code: “188768”	
Type as String: “1214000” MDER FLOAT: “030004BE”	Units as string: “joules” Units as code: “266112”
Timestamp: “20150504135813.22-0400”	

Editor Note: Impedance

Type as String: “Impedance” Type as Code: “188780”	
Type as String: “4567.8” MDER FLOAT: “FF00B26E”	Units as string: “joules” Units as code: “266432”
Timestamp: “20150504135813.22-0400”	

6.6 Blood Pressure Specific Functional Requirements

The following requirements outline the blood pressure specific set of options that Blood Pressure Plug-Ins implement. The Blood Pressure Plug-In technical specifications will address the necessary functions for support of these options. This device typically would be what one calls a 1 – N shot device where N is less than 25. However, if the device stores data persistently, the number of measurements could be very large.

> **Editor's note:**
>
> This section should specify all the detailed and necessary requirements that are specific to the profile, so that the architecture and technical specifications can be developed.

Blood Pressure devices supported by this plug in specification are expected to be able to report the systolic, diastolic, and optionally MAP components of the blood pressure and optionally the pulse rate. The description of the measurements reported by the plug in follows the IEEE 11073 10407 Blood Pressure specialization specification but that does not mean the device itself must follow that specification. However the device must provide to the plug in the necessary information such that the plug can fulfil its reporting requirements as specified in this document.

The IEEE specification reports the blood pressure in what is referred to as a compound attribute; that is the attribute consists of multiple values. Whenever an IEEE device uses a compound attribute it must also have a metric id list attribute which tells, with a one-to-one correspondence, what each compound entry is. For example, the metric-id-list might contain the MDC codes for ‘systolic’, ‘diastolic’, and ‘MAP’. The compound basic nu observed value attribute might then contain ‘120’, ‘80’, ‘100’.

Label	Description	Release
BP-HSF-00	The Plug-In SHALL provide values as both strings (human consumption) and MDC codes for detailed understanding and machine processing).	1.0

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>BP-HSF-01</td><td>The Blood Pressure Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Blood Pressure object as defined in IEEE 11073 10407 Table 5.
Note: <b>Table 5 —Systolic/diastolic/MAP compound numeric object attributes</b>
<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th><th colspan="2">Standard configuration</th></tr>
<tr><th>Value</th><th>Qual</th><th>Value</th><th>Qual</th></tr>
<tr><td>Type</td><td>MDC_PART_SCADA |
MDC_PRESS_BLD_NONINV.</td><td>M</td><td></td><td>M</td></tr>
<tr><td>Metric-Id</td><td>See IEEE Std 11073-20601.</td><td><b>M</b></td><td>Attribute not initially present. If present
follow IEEE Std 11073-20601.
Note: MDC_PRESS_BLD_NONINV_SYS,
MDC_PRESS_BLD_NONINV_DIA, then
MDC_PRESS_BLD_NONINV_MEAN.</td><td><b>M</b></td></tr>
<tr><td>Compound-Basic-
Nu-Observed-Value</td><td>See IEEE Std 11073-20601.</td><td>M</td><td>If fixed format is used and the standard
configuration is not adjusted, this attribute
is mandatory; otherwise, the conditions
from IEEE Std 11073-20601 apply.</td><td>M</td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_MMHG or
MDC_DIM_KILO_PASCAL</td><td>M</td><td>MDC_DIM_MMHG.</td><td>M</td></tr>
</table></td><td>1.0</td></tr>
<tr><td>BP-HSF-01.1</td><td>The Blood Pressure Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit and 16 bit code; partition: code). <i>[The TYPE attribute value for the Blood Pressure is fixed for all BP devices]</i>
Example: String: “Non invasive blood pressure”
Code: “150020”
[MDC_PRESS_BLD_NONINV]</td><td>1.0</td></tr>
<tr><td>BP-HSF-01.2</td><td>The Blood Pressure Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07.
Example: 20150430045532.32-0400</td><td>1.0</td></tr>
<tr><td>BP-HSF-01.3</td><td>The Blood Pressure Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code).
Exasmple: String: “mm/Hg”
Code: “266016”
[MDC_DIM_MMHG]</td><td>1.0</td></tr>
</table>

Label	Description	Release
BP-HSF-01.4	The Blood Pressure Plug-In SHALL report the value reported from Metric Id attribute in two formats, i.e., float and string: 1- Float that represents the Blood Pressure report data measured by the targeted Blood Pressure and represented in a float format. This is for general use of application programming. 2- String that represents the Blood Pressure report data measured by the targeted Blood Pressure and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT. Example: String: “systolic” Code: “150021” [MDC_PRESS_BLD_NONINV_SYS] String: “108” MDER FLOAT: “0000006C” String: “diastolic” Code: “150022” [MDC_PRESS_BLD_NONINV_DIA] String: “63” MDER FLOAT: “0000003F” String: “diastolic” Code: “150023” [MDC_PRESS_BLD_NONINV_MEAN] String: “63” MDER FLOAT: “00000055”	1.0

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>BP-HSF-02</td><td>The Blood Pressure Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Pulse Rate object as defined in IEEE 11073 10407 (Table 6) <b><i>if</i></b> the device reports a pulse rate.
Note: <b>Table 6 —Pulse rate numeric object attributes</b>
<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th><th colspan="2">Standard configuration</th></tr>
<tr><th>Value</th><th>Qual</th><th>Value</th><th>Qual</th></tr>
<tr><td>Type</td><td>MDC_PART_SCADA |
MDC_PULS_RATE_NON_INV.</td><td>M</td><td>MDC_PART_SCADA |
MDC_PULS_RATE_NON_INV.</td><td>M</td></tr>
<tr><td>Metric-Id</td><td>See IEEE Std 11073-20601.</td><td><b>M</b></td><td>Attribute not initially present. If present
follow IEEE Std 11073-20601.</td><td><b>M</b></td></tr>
<tr><td>Compound-Basic-
Nu-Observed-Value</td><td></td><td>M</td><td></td><td>M</td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_BEAT_PER_MIN.</td><td>M</td><td>MDC_DIM_BEAT_PER_MIN.</td><td>M</td></tr>
</table></td><td>1.0</td></tr>
<tr><td>BP-HSF-02.1</td><td>If the pulse rate is reported, the Blood Pressure Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit and 16 bit code; partition:code). <i>[The TYPE attribute value for the Pulse Rate is fixed for all BP devices]</i>
Example: String: “Pulse Rate”
Code: “149546”
[MDC_PULS_RATE_NON_INV]</td><td>1.0</td></tr>
<tr><td>BP-HSF-02.2</td><td>If the pulse rate is reported, the Blood Pressure Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07.
Example: 20150430045532.32-0400</td><td>1.0</td></tr>
<tr><td>BP-HSF-02.3</td><td>If the pulse rate is reported, the Blood Pressure Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). <i>[The Unit code attribute value for the Pulse Rate is fixed for all BP devices]</i>
<i>Example:</i> String: “beats per min”
Code: “264864”
[MDC_DIM_BEAT_PER_MIN]</td><td>1.0</td></tr>
</table>

Label	Description	Release
BP-HSF-02.4	If the pulse rate is reported, the Blood Pressure Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string: 1- Float that represents the pulse rate report data measured by the targeted pulse rate and represented in a float format. This is for general use of application programming. 2- String that represents the pulse rate report data measured by the targeted pulse rate and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT. Example: String: “43” MDER FLOAT: “0000002B”	1.0

Editor Note: Blood Pressure report (note: we report the diastolic pressure and the mean of the diastolic)

Type as String: “Non invasive blood pressure” Type as Code: “150020”		
Type as String: “systolic” Type as Code: “150021”	Value as string: “108” Value as FLOAT: “0000006C”	Units as string: “mm/Hg” Units as code” “266016” millimetres of mercury"
Type as String: “diastolic” Type as Code: “150022”	Value as String: “63” Value as FLOAT: “0000003F”	Units as string: “mm/Hg” Units as code” “266016”
Type as String: “diastolic” Type as Code: “150023”	Value as String: “63” Value as FLOAT: “00000055	Units as string: “mm/Hg” Units as code” “266016
Type as String: “Timestamp” Value as String “20150504135813.22-0400”		

Note: Pulse Rate if present

Editor Note: Blood Pressure report (note: we report the diastolic pressure and the mean of the diastolic)

Type as String: “Pulse Rate” Type as Code: Code: “149546”	Value as string: “43” Value as FLOAT: “0000002B”	Units as string: “beats per min” Units as code: “264864”
Type as String: “Timestamp” Value as String “20150504135813.22-0400		

6.7 Glucometer Specific Functional Requirements

The following requirements outline glucometer specific set of options that Glucometer Plug-Ins implement. The Glucometer Plug-In technical specifications will address the necessary functions for support of these options. This device typically would be what one calls a storage device, thus the number of measurements could be very large.

Editor's note:

This section should specify all the detailed and necessary requirements that are specific to the profile, so that the architecture and technical specifications can be developed.

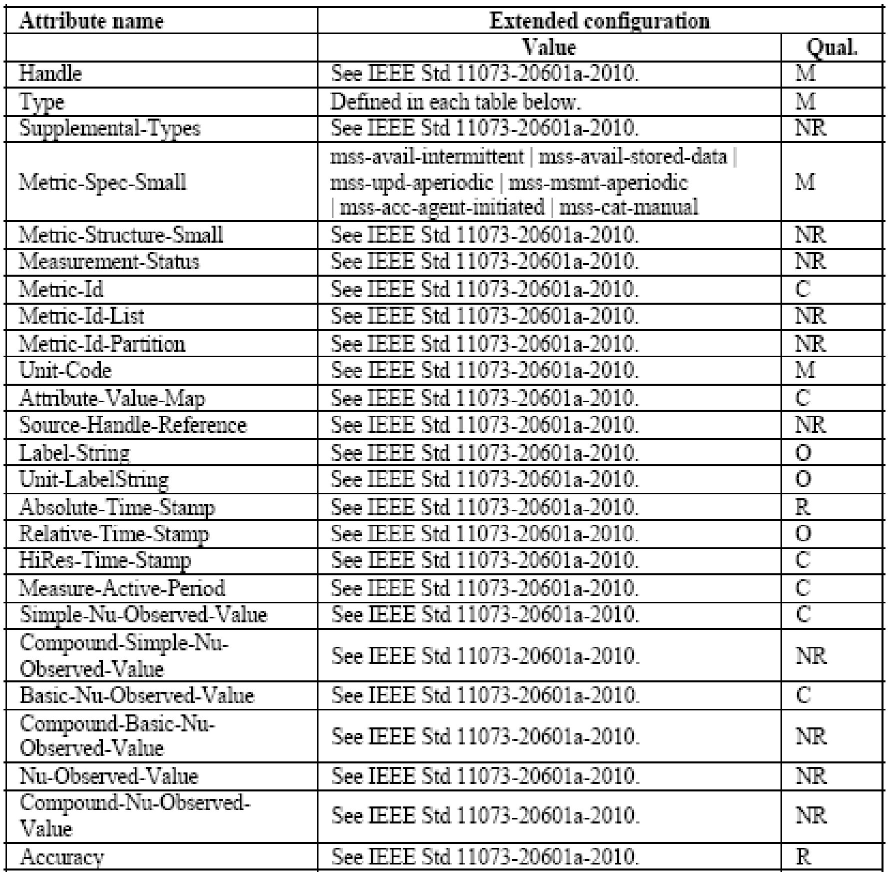
Glucometers are most frequently used off line. These devices need to be carried around with the individual essentially all the time. Thus most glucometers store measurement data in non-volatile persistent storage to be uploaded as needed with an extra action required to delete the data (temporarily stored data is auto deleted upon upload by spec). Consequently at upload, the number of measurements could be quite large. Uploads of persistent storage typically requires initiation by the collector of the data and is not done by the device. ‘Live’ and temporarily stored data is typically uploaded unsolicited by the device as soon as a connection is established.

In addition to the glucose concentration, glucometers may also report what is known as context measurements, for example when, relative to eating, the measurement was taken, what state of health one was in, the intensity of exercise activity, the medication one is on (relative to glucose control), etc. Glucometers may also measure Hb1Ac (glycated hemoglobin) which measures the average levels of blood sugar levels over the last three or so months. Glucometers supported by this plug in specification are expected to be able to report the Glucose concentration in any of several possible blood samples (plasma, whole blood, arterial, capillary, etc.) and or a control solution. The plug in is also expected to report Hb1Ac measurements and certain context measurements if the device supports them. The description of the measurements reported by the plug in follows the IEEE 11073 10417 Glucose specialization specification but that does not mean the device itself must follow that specification. However the device must provide to the plug in the necessary information such that the plug can fulfil its reporting requirements as specified in this document.

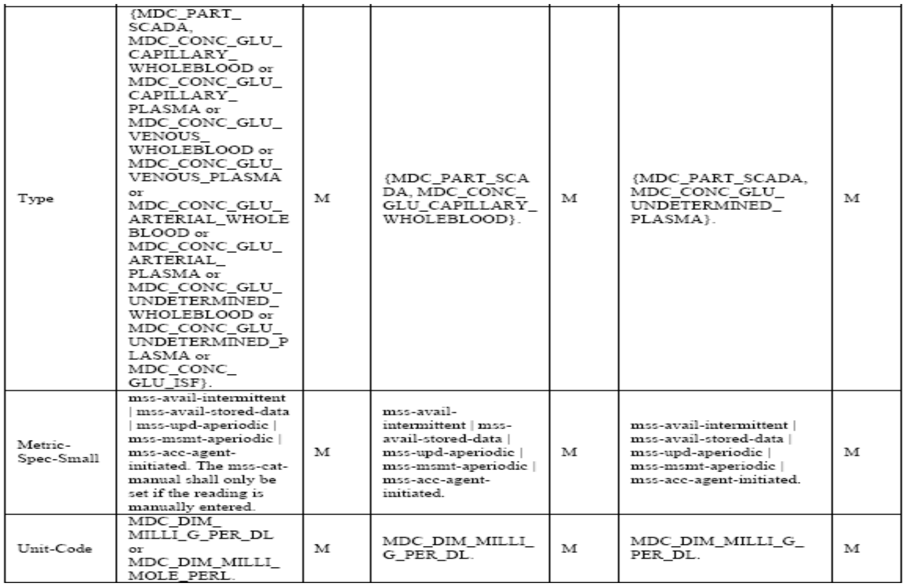
Label	Description	Release
GL-HSF-00	The Plug-In SHALL provide values as both strings (human consumption) and MDC codes for detailed understanding and machine processing).	1.0

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>GL-HSF-01</td><td>
The Glucose Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Glucose concentration and Glucose Control Solution objects as defined in IEEE 11073 10417 Tables 6, 7 and 12. (Note that the measurements reported by the Glucose concentration and Glucose control solution objects are identical except for the TYPE value.)

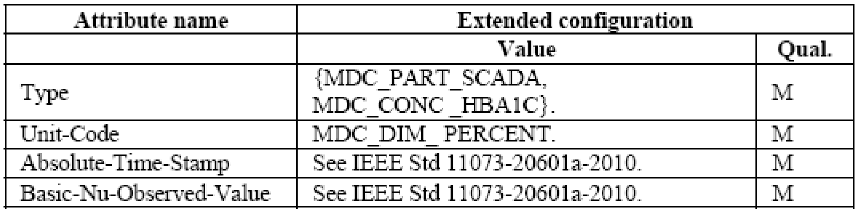
Table 6—Common glucose numeric object attributes



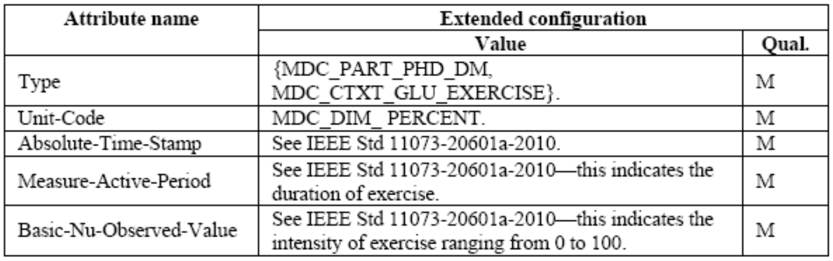
<table>
<tr><th>Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th></th><th>Value</th><th>Qual.</th></tr>
<tr><td>Handle</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td></tr>
<tr><td>Type</td><td>Defined in each table below.</td><td>M</td></tr>
<tr><td>Supplemental-Types</td><td>See IEEE Std 11073-20601a-2010.</td><td>NR</td></tr>
<tr><td>Metric-Spec-Small</td><td>mss-avail-intermittent | mss-avail-stored-data | mss-upd-aperiodic | mss-msmt-aperiodic | mss-acc-agent-initiated | mss-cat-manual</td><td>M</td></tr>
<tr><td>Metric-Structure-Small</td><td>See IEEE Std 11073-20601a-2010.</td><td>NR</td></tr>
<tr><td>Measurement-Status</td><td>See IEEE Std 11073-20601a-2010.</td><td>NR</td></tr>
<tr><td>Metric-Id</td><td>See IEEE Std 11073-20601a-2010.</td><td>C</td></tr>
<tr><td>Metric-Id-List</td><td>See IEEE Std 11073-20601a-2010.</td><td>NR</td></tr>
<tr><td>Metric-Id-Partition</td><td>See IEEE Std 11073-20601a-2010.</td><td>NR</td></tr>
<tr><td>Unit-Code</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td></tr>
<tr><td>Attribute-Value-Map</td><td>See IEEE Std 11073-20601a-2010.</td><td>C</td></tr>
<tr><td>Source-Handle-Reference</td><td>See IEEE Std 11073-20601a-2010.</td><td>NR</td></tr>
<tr><td>Label-String</td><td>See IEEE Std 11073-20601a-2010.</td><td>O</td></tr>
<tr><td>Unit-LabelString</td><td>See IEEE Std 11073-20601a-2010.</td><td>O</td></tr>
<tr><td>Absolute-Time-Stamp</td><td>See IEEE Std 11073-20601a-2010.</td><td>R</td></tr>
<tr><td>Relative-Time-Stamp</td><td>See IEEE Std 11073-20601a-2010.</td><td>O</td></tr>
<tr><td>HiRes-Time-Stamp</td><td>See IEEE Std 11073-20601a-2010.</td><td>C</td></tr>
<tr><td>Measure-Active-Period</td><td>See IEEE Std 11073-20601a-2010.</td><td>C</td></tr>
<tr><td>Simple-Nu-Observed-Value</td><td>See IEEE Std 11073-20601a-2010.</td><td>C</td></tr>
<tr><td>Compound-Simple-Nu-Observed-Value</td><td>See IEEE Std 11073-20601a-2010.</td><td>NR</td></tr>
<tr><td>Basic-Nu-Observed-Value</td><td>See IEEE Std 11073-20601a-2010.</td><td>C</td></tr>
<tr><td>Compound-Basic-Nu-Observed-Value</td><td>See IEEE Std 11073-20601a-2010.</td><td>NR</td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601a-2010.</td><td>NR</td></tr>
<tr><td>Compound-Nu-Observed-Value</td><td>See IEEE Std 11073-20601a-2010.</td><td>NR</td></tr>
<tr><td>Accuracy</td><td>See IEEE Std 11073-20601a-2010.</td><td>R</td></tr>
</table>

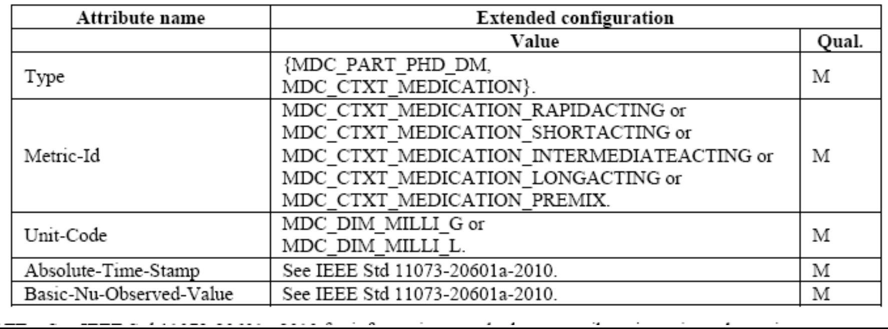


<table>
<tr><td>Type</td><td>{MDC_PART_SCADA, MDC_CONC_GLU_CAPILLARY_WHOLEBLOOD or MDC_CONC_GLU_CAPILLARY_PLASMA or MDC_CONC_GLU_VENOUS_WHOLEBLOOD or MDC_CONC_GLU_VENOUS_PLASMA or MDC_CONC_GLU_ARTERIAL_WHOLEBLOOD or MDC_CONC_GLU_ARTERIAL_PLASMA or MDC_CONC_GLU_UNDETERMINED_WHOLEBLOOD or MDC_CONC_GLU_UNDETERMINED_PLASMA or MDC_CONC_GLU_ISF}.</td><td>M</td><td>{MDC_PART_SCADA, MDC_CONC_GLU_CAPILLARY_WHOLEBLOOD}.</td><td>M</td><td>{MDC_PART_SCADA, MDC_CONC_GLU_UNDETERMINED_PLASMA}.</td><td>M</td></tr>
<tr><td>Metric-Spec-Small</td><td>mss-avail-intermittent | mss-avail-stored-data | mss-upd-aperiodic | mss-msmt-aperiodic | mss-acc-agent-initiated. The mss-cat-manual shall only be set if the reading is manually entered.</td><td>M</td><td>mss-avail-intermittent | mss-avail-stored-data | mss-upd-aperiodic | mss-msmt-aperiodic | mss-acc-agent-initiated.</td><td>M</td><td>mss-avail-intermittent | mss-avail-stored-data | mss-upd-aperiodic | mss-msmt-aperiodic | mss-acc-agent-initiated.</td><td>M</td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_MILLI_G_PER_DL or MDC_DIM_MILLI_MOLE_PERL.</td><td>M</td><td>MDC_DIM_MILLI_G_PER_DL.</td><td>M</td><td>MDC_DIM_MILLI_G_PER_DL.</td><td>M</td></tr>
</table>
</td><td>1.0</td></tr>
</table>

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>GL-HSF-01.1</td><td>The Glucose Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition: code). These type values indicate the concentration measured in the medium; plasma, whole blood, arterial plasma, control solution, venous plasma, etc.
Example:
String: “Glucose concentration”
Code: “160368”
[MDC_CONC_GLU_UNDETERMINED_PLASMA]</td><td>1.0</td></tr>
<tr><td>GL-HSF-01.2</td><td>The Glucose Plug-In SHALL report the concentration value reported from the appropriate *-Nu-Observed-Val in two formats, i.e., float and string:
1- Float that represents the Glucose report data measured by the targeted Glucose meter and represented in a float format. This is for general use of application programming.
2- String that represents the Glucose report data measured by the targeted Glucose Meter and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT.
Example
String: “106”
MDER FLOAT: “0000006A”</td><td>1.0</td></tr>
<tr><td>GL-HSF-01.3</td><td>The Glucose Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code).
MDC_DIM_MILLI_G_PER_DL Glucose Unit Value Unit Description “ mg/dL “
MDC_DIM_MILLI_MOLE_PER_L Glucose Unit Value Unit Description “ mmol/L”
Example:
String: “mg/dl”
Code: “264274”
[MDC_DIM_MILLI_G_PER_DL]</td><td>1.0</td></tr>
<tr><td>GL-HSF-01.4</td><td>The Glucose Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07.
Example:
Timestamp: 20150422080512.00-0400</td><td>1.0</td></tr>
<tr><td>GL-HSF-02</td><td>The Glucose Plug-In SHALL report the values as stated in these guidelines for the equivalent of the HbA1c object as defined in IEEE 11073 10417 Tables 6 and 8 if the device reports such a measurement.
Table 8—HbA1c numeric object attributes
<table><tr><th>Attribute name</th><th colspan="2">Extended configuration</th></tr><tr><th></th><th>Value</th><th>Qual.</th></tr><tr><td>Type</td><td>{MDC_PART_SCADA, MDC_CONC _HBA1C}.</td><td>M</td></tr><tr><td>Unit-Code</td><td>MDC_DIM_ PERCENT.</td><td>M</td></tr><tr><td>Absolute-Time-Stamp</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td></tr><tr><td>Basic-Nu-Observed-Value</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td></tr></table></td><td>1.0</td></tr>
</table>

Label	Description	Release
GL-HSF-02.1	If the device reports an HbA1c measurement, the Glucose Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). *[The TYPE attribute value for the HbA1c is fixed for all Glucometer devices]* Example: String: “HbA1c level” Code: “160220” [MDC_CONC_HBA1C]	1.0
GL-HSF-02.2	If the device reports an HbA1c measurement, the Glucose Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string: 1- Float that represents an HbA1c measurement report data measured by the targeted Glucose meter and represented in a float format. This is for general use of application programming. 2- String that represents an HbA1c measurement data measured by the targeted Glucose Meter and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT. Example: String: “4.2” MDER FLOAT: “FF00002A”	1.0
GL-HSF-02.3	If the device reports an HbA1c measurement, the Glucose Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). *[The Unit code attribute value for the HbA1c is fixed for all Glucometer devices]* Example: String: “%” Code: “262688” [MDC_DIM_PERCENT]	1.0
GL-HSF-02.4	If the device reports an HbA1c measurement, the Glucose Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07. Timestamp:20150422080512.00-0400	1.0

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>GL-HSF-03</td><td>The Glucose Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Context exercise object as defined in IEEE 11073 10417 Tables 6 and 9 if the device reports such a measurement.
Table 9—Context exercise numeric object attributes

<table>
<tr><th>Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th></th><th>Value</th><th>Qual.</th></tr>
<tr><td>Type</td><td>{MDC_PART_PHD_DM, MDC_CTXT_GLU_EXERCISE}.</td><td>M</td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_ PERCENT.</td><td>M</td></tr>
<tr><td>Absolute-Time-Stamp</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td></tr>
<tr><td>Measure-Active-Period</td><td>See IEEE Std 11073-20601a-2010—this indicates the duration of exercise.</td><td>M</td></tr>
<tr><td>Basic-Nu-Observed-Value</td><td>See IEEE Std 11073-20601a-2010—this indicates the intensity of exercise ranging from 0 to 100.</td><td>M</td></tr>
</table></td><td>1.0</td></tr>
<tr><td>GL-HSF-03.1</td><td>If the device reports a context exercise measurement (which is a measure of the exercise intensity), the Glucose Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). <i>[The TYPE attribute value for the Context Exercise is fixed for all Glucometer devices]</i>
Example:
String: “context exercise”
Code: “160220”
{MDC_PART_PHD_DM, MDC_CTXT_GLU_EXERCISE}.</td><td>1.0</td></tr>
<tr><td>GL-HSF-03.2</td><td>If the device reports a context exercise measurement, the Glucose Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string:
1- Float that represents a context exercise measurement report data measured by the targeted Glucose meter and represented in a float format. This is for general use of application programming.
2- String that represents a context exercise measurement data measured by the targeted Glucose Meter and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT.
Example:
String: “50”
MDER FLOAT: “FF00002A”</td><td>1.0</td></tr>
<tr><td>GL-HSF-03.3</td><td>If the device reports a context exercise measurement, the Glucose Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). <i>[The Unit code attribute value for the a context exercise is fixed for all Glucometer devices]</i>
Example:
String: “%”
Code: “262688”
[MDC_DIM_PERCENT]</td><td>1.0</td></tr>
</table>

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>GL-HSF-03.4</td><td>If the device reports a context exercise measurement, the Glucose Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07.
Timestamp:20150422080512.00-0400</td><td>1.0</td></tr>
<tr><td>GL-HSF-04</td><td>The Glucose Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Context medication object as defined in IEEE 11073 10417 Tables 6 and 10 if the device reports such a measurement.
Table 10 —Context medication numeric object attributes
<table>
<tr><th>Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th></th><th>Value</th><th>Qual.</th></tr>
<tr><td>Type</td><td>{MDC_PART_PHD_DM,
MDC_CTXT_MEDICATION}.</td><td>M</td></tr>
<tr><td>Metric-Id</td><td>MDC_CTXT_MEDICATION_RAPIDACTING or
MDC_CTXT_MEDICATION_SHORTACTING or
MDC_CTXT_MEDICATION_INTERMEDIATEACTING or
MDC_CTXT_MEDICATION_LONGACTING or
MDC_CTXT_MEDICATION_PREMIX.</td><td>M</td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_MILLI_G or
MDC_DIM_MILLI_L.</td><td>M</td></tr>
<tr><td>Absolute-Time-Stamp</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td></tr>
<tr><td>Basic-Nu-Observed-Value</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td></tr>
</table></td><td>1.0</td></tr>
<tr><td>GL-HSF-04.1</td><td>If the device reports a context medication measurement, the Glucose Plug-In SHALL report the value of the TYPE* attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). *If the Metric Id is used the Glucose Plug-In SHALL replace the code value with this value and if the Metric-Id-partition is present the partition value SHALL be replaced with this value.
Example:
String: “context medication”
Code: “160220”
{MDC_PART_PHD_DM, MDC_CTXT_MEDICATION}.</td><td>1.0</td></tr>
<tr><td>GL-HSF-04.2</td><td>If the device reports a context medication measurement, the Glucose Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string:
1- Float that represents context medication measurement report data measured by the targeted Glucose meter and represented in a float format. This is for general use of application programming.
2- String that represents a context medication measurement data measured by the targeted Glucose Meter and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT.
Example:
String: “10”
MDER FLOAT: “FF00002A</td><td>1.0</td></tr>
</table>

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>GL-HSF-04.3</td><td>If the device reports a context medication measurement, the Glucose Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code).
MDC_DIM_MILLI_G_PER_DL Glucose Unit Value Unit Description “ mg/dL “
MDC_DIM_MILLI_MOLE_PER_L Glucose Unit Value Unit Description “ mmol/L”
MDC_DIM_MILLI_G
OR MDC_DIM_MILLI_L
Example:
String: “mg/dL”
Code: “160220”
Note:
<b>Diabetapedia definition</b>: Milligrams per deciliter, a unit of measure that shows the concentration of a substance in a specific amount of fluid. In the United States, blood glucose test results are reported as mg/dL. Medical journals and other countries use millimoles per liter (mmol/L). To convert to mg/dL from mmol/L, multiply mmol/L by 18. Example: 10 mmol/L, 18 = 180 mg/dL.</td><td>1.0</td></tr>
<tr><td>GL-HSF-04.4</td><td>If the device reports a context medication measurement, the Glucose Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07.
Example:
Timestamp:20150422080512.00-0400</td><td>1.0</td></tr>
<tr><td>GL-HSF-05</td><td>The Glucose Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Context Carbohydrates object as defined in IEEE 11073 10417 Tables 6 and 11 if the device reports such a measurement.

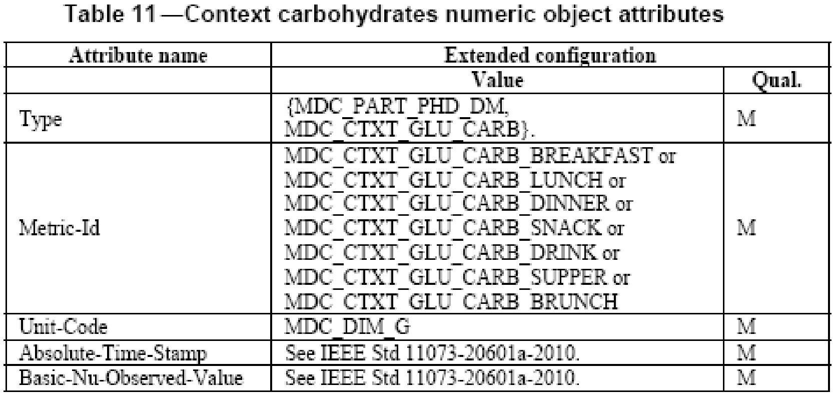
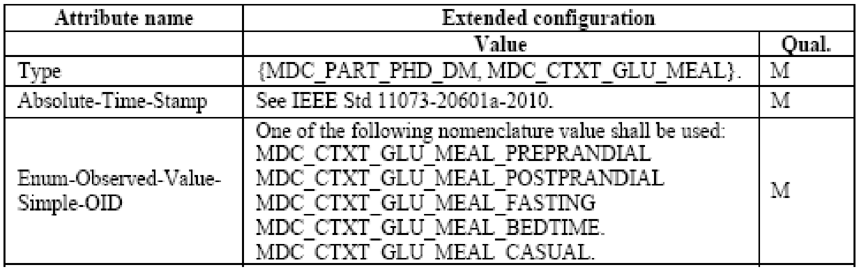



Table 11 —Context carbohydrates numeric object attributes
<table><tr><th>Attribute name</th><th colspan="2">Extended configuration</th></tr><tr><th></th><th>Value</th><th>Qual.</th></tr><tr><td>Type</td><td>{MDC_PART_PHD_DM, MDC_CTXT_GLU_CARB}.</td><td>M</td></tr><tr><td>Metric-Id</td><td>MDC_CTXT_GLU_CARB_BREAKFAST or MDC_CTXT_GLU_CARB_LUNCH or MDC_CTXT_GLU_CARB_DINNER or MDC_CTXT_GLU_CARB_SNACK or MDC_CTXT_GLU_CARB_DRINK or MDC_CTXT_GLU_CARB_SUPPER or MDC_CTXT_GLU_CARB_BRUNCH</td><td>M</td></tr><tr><td>Unit-Code</td><td>MDC_DIM_G</td><td>M</td></tr><tr><td>Absolute-Time-Stamp</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td></tr><tr><td>Basic-Nu-Observed-Value</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td></tr></table></td><td>1.0</td></tr>
<tr><td>GL-HSF-05.1</td><td>If the device reports a context carbohydrates measurement, the Glucose Plug-In SHALL report the value of the TYPE* attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). *If the Metric Id is used the Glucose Plug-In SHALL replace the code value with this value and if the Metric-Id-partition is present the partition value SHALL be replaced with this value.
Example:
String: “Breakfast Carbohydrates”
Code: “8417768”
[MDC_CTXT_GLU_CARB_BREAKFAST]</td><td>1.0</td></tr>
</table>

Label	Description	Release
GL-HSF-05.2	If the device reports a context carbohydrates measurement, the Glucose Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string: 1. Float that represents a context carbohydrates measurement reported data measured by the targeted Glucose meter and represented in a float format. This is for general use of application programming. 2. String that represents a context carbohydrates measurement data measured by the targeted Glucose Meter and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT. Example: String: “32” MDER FLOAT: “00000020”	1.0
GL-HSF-05.3	If the device reports a context carbohydrates measurement, the Glucose Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). *[The Unit code attribute value for the a context exercise is fixed for all Glucometer devices]* Example: String: “g” Code: “263908” [MDC_DIM_G]	1.0
GL-HSF-05.4	If the device reports a context carbohydrates measurement, the Glucose Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07. Example: Timestamp:20150422080512.00-0400	1.0
GL-HSF-06	The Glucose Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Context Meal object as defined in IEEE 11073 10417 Tables 13 and 16 if the device reports such a measurement. Table 16—Context meal enumeration object attributes 	1.0

Attribute name	Extended configuration	
	Value	Qual.
Type	{MDC_PART_PHD_DM, MDC_CTXT_GLU_MEAL}.	M
Absolute-Time-Stamp	See IEEE Std 11073-20601a-2010.	M
Enum-Observed-Value-Simple-OID	One of the following nomenclature value shall be used: MDC_CTXT_GLU_MEAL_PREPRANDIAL MDC_CTXT_GLU_MEAL_POSTPRANDIAL MDC_CTXT_GLU_MEAL_FASTING MDC_CTXT_GLU_MEAL_BEDTIME. MDC_CTXT_GLU_MEAL_CASUAL.	M

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>GL-HSF-06.1</td><td>If the device reports a context meal measurement, the Glucose Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). <i>[The TYPE attribute value for the Context Meal is fixed for all Glucometer devices]</i>
Example:
String: “Context Meal”
Code: “8417864”
[MDC_CTXT_GLU_MEAL]</td><td>1.0</td></tr>
<tr><td>GL-HSF-06.2</td><td>If the device reports a context meal value, the Glucose Plug-In SHALL report the value reported from the Enum-Observed-Value-Simple-OID as a human readable string and as its 32-bit MDC code; combine the 16-bit partition and 16 bit code; partition:code. (Note there is no unit code associated with this measurement.)
Example:
String: “Casual”
Code: “8417880”
[MDC_CTXT_GLU_MEAL_CASUAL]</td><td>1.0</td></tr>
<tr><td>GL-HSF-06.3</td><td>If the device reports a context meal value, the Glucose Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07.
Example:
Timestamp:20150422080512.00-0400</td><td>1.0</td></tr>
<tr><td>GL-HSF-07</td><td>The Glucose Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Context Sample Location object as defined in IEEE 11073 10417 Tables 13 and 17 if the device reports such a measurement.

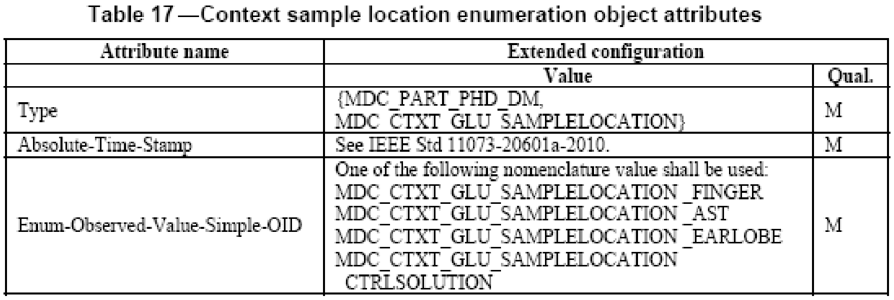



Table 17 —Context sample location enumeration object attributes
<table>
<tr><th>Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th></th><th>Value</th><th>Qual.</th></tr>
<tr><td>Type</td><td>{MDC_PART_PHD_DM,
MDC_CTXT_GLU_SAMPLELOCATION}</td><td>M</td></tr>
<tr><td>Absolute-Time-Stamp</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td></tr>
<tr><td>Enum-Observed-Value-Simple-OID</td><td>One of the following nomenclature value shall be used:
MDC_CTXT_GLU_SAMPLELOCATION _FINGER
MDC_CTXT_GLU_SAMPLELOCATION _AST
MDC_CTXT_GLU_SAMPLELOCATION _EARLOBE
MDC_CTXT_GLU_SAMPLELOCATION _CTRLSOLUTION</td><td>M</td></tr>
</table></td><td>1.0</td></tr>
<tr><td>GL-HSF-07.1</td><td>If the device reports a context sample location measurement, the Glucose Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). <i>[The TYPE attribute value for the Context Meal is fixed for all Glucometer devices]</i>
Example:
String: “Context Sample Location”</td><td>1.0</td></tr>
<tr><td>GL-HSF-07.2</td><td>If the device reports a context sample location measurement, the Glucose Plug-In SHALL report the value reported from the Enum-Observed-Value-Simple-OID as a human readable string and as its 32-bit MDC code; combine the 16-bit partition and 16 bit code; partition:code. (Note there is no unit code associated with this measurement.)
Example:
String: “Finger”
Code: “8417880”</td><td>1.0</td></tr>
<tr><td>GL-HSF-07.3</td><td>If the device reports a context sample location measurement, the Glucose Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07.
Example:
Timestamp:20150422080512.00-0400</td><td>1.0</td></tr>
</table>

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>GL-HSF-08</td><td>The Glucose Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Context Tester object as defined in IEEE 11073 10417 Tables 13 and 18 if the device reports such a measurement.

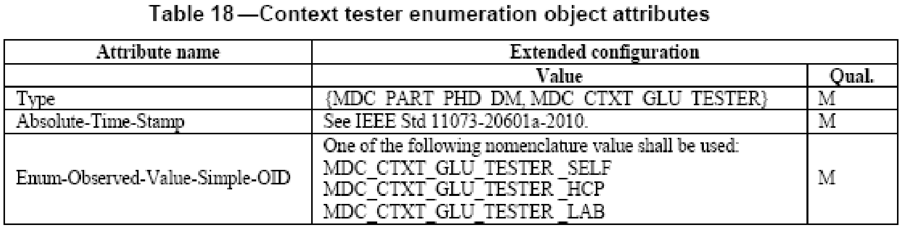


Table 18 —Context tester enumeration object attributes

<table>
<tr><th>Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th></th><th>Value</th><th>Qual.</th></tr>
<tr><td>Type</td><td>{MDC_PART_PHD_DM, MDC_CTXT_GLU_TESTER}</td><td>M</td></tr>
<tr><td>Absolute-Time-Stamp</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td></tr>
<tr><td>Enum-Observed-Value-Simple-OID</td><td>One of the following nomenclature value shall be used:
MDC_CTXT_GLU_TESTER _SELF
MDC_CTXT_GLU_TESTER _HCP
MDC_CTXT_GLU_TESTER _LAB</td><td>M</td></tr>
</table>
</td><td>1.0</td></tr>
<tr><td>GL-HSF-08.1</td><td>If the device reports a context tester measurement, the Glucose Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). <i>[The TYPE attribute value for the Context Meal is fixed for all Glucometer devices]</i>
Example:
String “Context Tester”
Code: “???”</td><td>1.0</td></tr>
<tr><td>GL-HSF-08.2</td><td>If the device reports a context tester measurement, the Glucose Plug-In SHALL report the value reported from the Enum-Observed-Value-Simple-OID as a human readable string and as its 32-bit MDC code; combine the 16-bit partition and 16 bit code; partition:code. (Note there is no unit code associated with this measurement.)
Example:
String: “Self”
Code: “8417880”</td><td>1.0</td></tr>
<tr><td>GL-HSF-08.3</td><td>If the device reports a context tester measurement, the Glucose Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07.
Example: Timestamp:20150422080512.00-0400</td><td>1.0</td></tr>
<tr><td>GL-HSF-09</td><td>The Glucose Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Context Health object as defined in IEEE 11073 10417 Tables 13 and 19 if the device reports such a measurement.

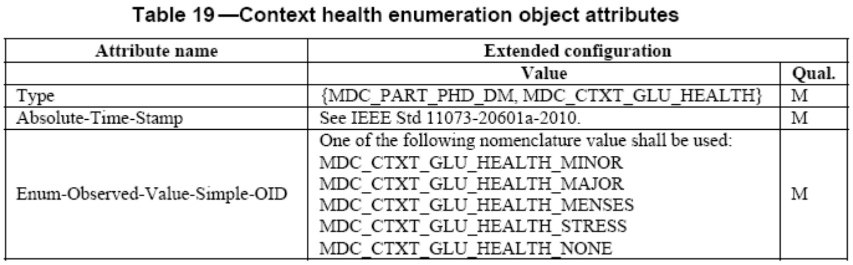


Table 19 —Context health enumeration object attributes

<table>
<tr><th>Attribute name</th><th colspan="2">Extended configuration</th></tr>
<tr><th></th><th>Value</th><th>Qual.</th></tr>
<tr><td>Type</td><td>{MDC_PART_PHD_DM, MDC_CTXT_GLU_HEALTH}</td><td>M</td></tr>
<tr><td>Absolute-Time-Stamp</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td></tr>
<tr><td>Enum-Observed-Value-Simple-OID</td><td>One of the following nomenclature value shall be used:
MDC_CTXT_GLU_HEALTH_MINOR
MDC_CTXT_GLU_HEALTH_MAJOR
MDC_CTXT_GLU_HEALTH_MENSES
MDC_CTXT_GLU_HEALTH_STRESS
MDC_CTXT_GLU_HEALTH_NONE</td><td>M</td></tr>
</table>
</td><td>1.0</td></tr>
<tr><td>GL-HSF-09.1</td><td>If the device reports a context health measurement, the Glucose Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). <i>[The TYPE attribute value for the Context Meal is fixed for all Glucometer devices]</i>
Example:
String: “Context Health”
Code: “8417880”</td><td>1.0</td></tr>
</table>

Label	Description	Release
GL-HSF-09.2	If the device reports a context health measurement, the Glucose Plug-In SHALL report the value reported from the Enum-Observed-Value-Simple-OID as a human readable string and as its 32-bit MDC code; combine the 16-bit partition and 16 bit code; partition:code. (Note there is no unit code associated with this measurement.) Example: String: “Minor” Code: “8417880”	1.0
GL-HSF-09.3	If the device reports a context health measurement, the Glucose Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07. Example: Timestamp:20150422080512.00-0400	1.0

Editor Note: Glucose concentration

Type as String: “Glucose concentration” Type as Code: “160368”	
Type as String: “106” MDER FLOAT: “0000006A”	Units as string: “mg/dl” Units as code: “264274”
Timestamp: “20150504135813.22-0400”	

Editor Note: Glucose Control Solution

Type as String: “HbA1c level” Type as Code: “160220”	
Type as String: “4.2” MDER FLOAT: “FF00002A”	Units as string: “mg/dl” Units as code “264274”
Timestamp: “20150504135813.22-0400”	

Editor Note: Context exercise

Type as String: “context exercise” Type as Code: “160220”	
Type as String: “50” MDER FLOAT: “FF00002A”	Units as string: “%” Units as code Code: “262688”
Timestamp: “20150504135813.22-0400”	

Editor Note: Context medication

Type as String: “context medication” Type as Code: “160220”	
Type as String: “10” MDER FLOAT: “FF00002A”	Units as string: “mg/dL” Units as code: “160220”
Timestamp: “20150504135813.22-0400”	

Editor Note: Context Carbohydrates

Type as String: “Breakfast Carbohydrates” Type as Code: “8417768”	

Type as String: “32” MDER FLOAT: “00000020”	Units as string: “g” Units as code: “263908”
Timestamp: “20150504135813.22-0400”	

Editor Note: Context Meal object

Type as String: “Context Meal ” Type as Code: “8417864”
Type as String: “Casual” Units as code: “8417880”
Timestamp: “20150504135813.22-0400”

Editor Note: Context Sample Location

Type as String: “Context Sample Location” Type as Code: “8417768”
Type as String: “Finger” Units as code: “8417880”
Timestamp: “20150504135813.22-0400”

Editor Note: Context Tester

Type as String: “Context Tester” Type as Code: “8417768”
Type as String: “Self” Units as code: “8417880”
Timestamp: “20150504135813.22-0400”

Editor Note: Context Health

Type as String: “Context Health” Type as Code: “8417880”
Type as String: “Minor” Units as code: “8417880”
Timestamp: “20150504135813.22-0400”

6.8 Heart Rate / Electrocardiogram Specific Functional Requirements

The following requirements outline the heart rate specific set of options that Heart Rate Plug-Ins implement. The Heart Rate Plug-In technical specifications will address the necessary functions for support of these options. This device would typically stream measurements.

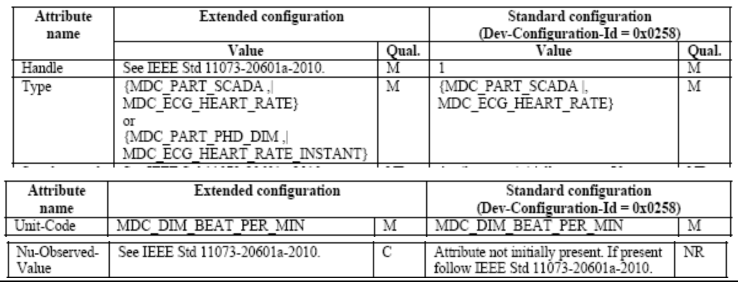
Editor's note:

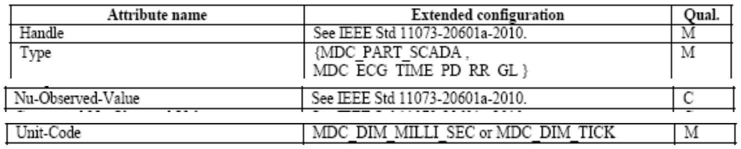
This section should specify all the detailed and necessary requirements that are specific to the profile, so that the architecture and technical specifications can be developed.

Heart Rate and Electrocardiogram (ECG) devices supported by this plug in specification are expected to be able to report the heart rate and if an RR interval is supported, those measurements are also to be reported. The description of the measurement reported by the plug in follows the IEEE 11073 10406 ECG Heart Rate sub profile specialization specification but that does not mean the device itself must follow that specification. However the device must provide to the plug in the necessary information such that the plug can fulfil its reporting requirements as specified in this document.

It should be noted that Bluetooth Low Energy supports the Heart Rate Profile which is modelled after the IEEE 11073 10406 specification. However, Bluetooth Low Energy heart rate monitors are likely to be for more casual use such as for exercising whereas IEEE devices are likely to be used in more medical care situations since the IEEE devices use electric sensor wires attached to the body whereas casual heart rate monitors use other techniques. The IEEE ECG specification supports a couple of sub profiles. The heart rate Plug-In defined here only supports the Heart Rate sub profile of the IEEE ECG specialization. Other sub profiles of the IEEE ECG specialization are not required to support the Heart Rate measurement.

Additional objects more oriented towards medical information are supported by the IEEE ECG heart rate sub profile. The Heart Rate Plug-In does not support those measurements. The Bluetooth Low Energy Heart Rate monitor may report a calories burned measurement. This measurement is NOT supported by the IEEE ECG specialization but is present in the IEEE 11073 10441 cardiovascular specialization specification. The Heart Rate Plug-In will report this measurement if the device supports it. In the future an IEEE 11073 10441 Cardiovascular heart rate sub profile may be defined which will be more in line with the casual use, Bluetooth Low Energy Heart Rate monitors.

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>HR-HSF-00</td><td>The Plug-In SHALL provide values as both strings (human consumption) and MDC codes for detailed understanding and machine processing).</td><td>1.0</td></tr>
<tr><td>HR-HSF-01</td><td>The Heart Rate Plug-In SHALL report the values as stated in these guidelines for the equivalent of the Heart Rate object as defined in IEEE 11073 10406 Table 6.
Table 6—Heart rate numeric object attributes

<table>
<tr><th rowspan="2">Attribute name</th><th colspan="2">Extended configuration</th><th colspan="2">Standard configuration
(Dev-Configuration-Id = 0x0258)</th></tr>
<tr><th>Value</th><th>Qual.</th><th>Value</th><th>Qual.</th></tr>
<tr><td>Handle</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td><td>1</td><td>M</td></tr>
<tr><td>Type</td><td>{MDC_PART_SCADA ,|
MDC_ECG_HEART_RATE}
or
{MDC_PART_PHD_DIM ,|
MDC_ECG_HEART_RATE_INSTANT}</td><td>M</td><td>{MDC_PART_SCADA |,
MDC_ECG_HEART_RATE}</td><td>M</td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_BEAT_PER_MIN</td><td>M</td><td>MDC_DIM_BEAT_PER_MIN</td><td>M</td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601a-2010.</td><td>C</td><td>Attribute not initially present. If present follow IEEE Std 11073-20601a-2010.</td><td>NR</td></tr>
</table></td><td>1.0</td></tr>
<tr><td>HR-HSF-01.1</td><td>The Heart Rate Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition: code).
Example:
String: “Heart rate”
Code: “147842”
[MDC_ECG_HEART_RATE]</td><td>1.0</td></tr>
</table>

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>HR-HSF-01.2</td><td>The Heart Rate Plug-In SHALL report the concentration value reported from the appropriate *-Nu-Observed-Val in two formats, i.e., float and string:
1- Float that represents the Heart Rate data measured by the targeted Heart Rate monitor and represented in a float format. This is for general use of application programming.
2- String that represents the Heart Rate report data measured by the targeted Heart Rate monitor and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT.
Example:
String: “75”
MDER FLOAT: “0000002D”</td><td>1.0</td></tr>
<tr><td>HR-HSF-01.3</td><td>The Heart Rate Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). <i>[The Unit code attribute value for the Heart Rate object is fixed for all heart rate devices.]</i>
Example:
String: “beats per min”
Code: “264864”
[MDC_DIM_BEAT_PER_MIN]</td><td>1.0</td></tr>
<tr><td>HR-HSF-01.4</td><td>The Heart Rate Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07. (Note that Bluetooth Low Energy heart rate monitors do not report any time stamp.)
Timestamp: 20150506164232.00-0400</td><td>1.0</td></tr>
<tr><td>HR-HSF-02</td><td>The Heart Rate Plug-In SHALL report the values as stated in these guidelines for the equivalent of the RR interval object as defined in IEEE 11073 10406 Table 7 if the device supports the measurement.
<b>Table 7—R-R interval numeric object attributes</b>

<table>
<tr><th>Attribute name</th><th>Extended configuration</th><th>Qual.</th></tr>
<tr><td>Handle</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td></tr>
<tr><td>Type</td><td>{MDC_PART_SCADA ,
MDC_ECG_TIME_PD_RR_GL }</td><td>M</td></tr>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601a-2010.</td><td>C</td></tr>
<tr><td>Unit-Code</td><td>MDC_DIM_MILLI_SEC or MDC_DIM_TICK</td><td>M</td></tr>
</table>
</td><td>1.0</td></tr>
</table>

Label	Description	Release
HR-HSF-02.1	If the device supports RR interval measurements, the Heart Rate Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). *[The TYPE code attribute value for the Heart Rate object is fixed for all heart rate devices.]* Example: String: “RR interval” Code: “147240” [MDC_ECG_TIME_PD_RR_GL]	1.0
HR-HSF-02.2	If the device supports RR interval measurements, the Heart Rate Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute as an MDER FLOAT and as string with appropriate precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Heart Rate Plug-In maps to an MDER FLOAT.) If the device supports RR interval measurements, the Heart Rate Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string: 1- Float that represents the Heart Rate report data measured by the targeted Heart Rate monitor and represented in a float format. This is for general use of application programming. 2- String that represents the Heart Rate report data measured by the targeted Heart Rate monitor and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT. Example: String: “1365” MDER FLOAT: “00000555”	1.0
HR-HSF-02.3	If the device supports RR interval measurements, the Heart Rate Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). The Unit code SHALL be in milliseconds (MDC_DIM_MILLI_SEC) and not in ‘ticks’. Example: String: “ms” Code: “264338” [MDC_DIM_MILLI_SEC]	1.0

<table>
<tr><th>Label</th><th>Description</th><th>Release</th></tr>
<tr><td>HR-HSF-02.4</td><td>If the device supports RR interval measurements, the Heart Rate Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07. (Note that Bluetooth Low Energy heart rate monitors do not report any time stamp. The Plug-In is responsible for computing the time stamp for each RR interval measurement reported, as Bluetooth Low Energy devices may send more than one RR interval in a characteristic and thus subsequent RR interval value timestamps will need to be computed.)
Example:
Timestamp: 20150506164232.00-0400</td><td>1.0</td></tr>
<tr><td>HR-HSF-03</td><td>The Heart Rate Plug-In SHALL report the values as stated in these guidelines for the equivalent of the energy expended object as defined in IEEE 11073 10441 Table x if the device supports the measurement. IEEE ECG devices will not support this measurement.

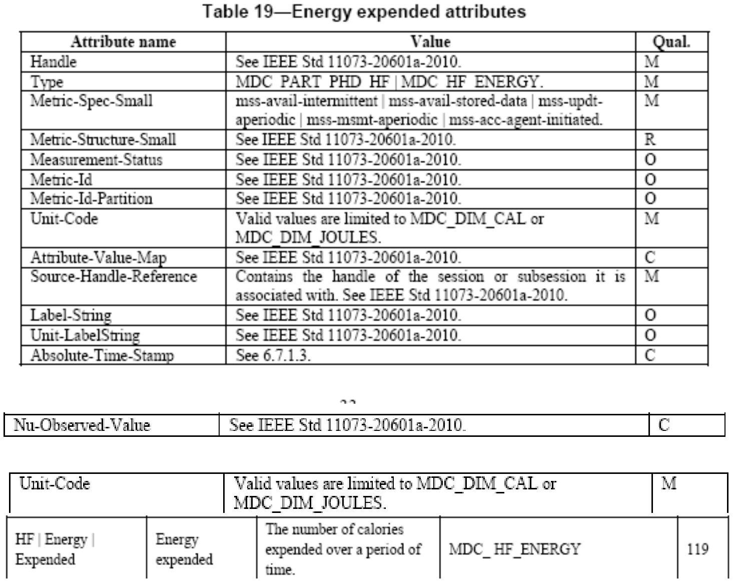



Table 19—Energy expended attributes
<table>
<tr><th>Attribute name</th><th>Value</th><th>Qual.</th></tr>
<tr><td>Handle</td><td>See IEEE Std 11073-20601a-2010.</td><td>M</td></tr>
<tr><td>Type</td><td>MDC_PART_PHD_HF | MDC_HF_ENERGY.</td><td>M</td></tr>
<tr><td>Metric-Spec-Small</td><td>mss-avail-intermittent | mss-avail-stored-data | mss-updt-aperiodic | mss-msmt-aperiodic | mss-acc-agent-initiated.</td><td>M</td></tr>
<tr><td>Metric-Structure-Small</td><td>See IEEE Std 11073-20601a-2010.</td><td>R</td></tr>
<tr><td>Measurement-Status</td><td>See IEEE Std 11073-20601a-2010.</td><td>O</td></tr>
<tr><td>Metric-Id</td><td>See IEEE Std 11073-20601a-2010.</td><td>O</td></tr>
<tr><td>Metric-Id-Partition</td><td>See IEEE Std 11073-20601a-2010.</td><td>O</td></tr>
<tr><td>Unit-Code</td><td>Valid values are limited to MDC_DIM_CAL or MDC_DIM_JOULES.</td><td>M</td></tr>
<tr><td>Attribute-Value-Map</td><td>See IEEE Std 11073-20601a-2010.</td><td>C</td></tr>
<tr><td>Source-Handle-Reference</td><td>Contains the handle of the session or subsession it is associated with. See IEEE Std 11073-20601a-2010.</td><td>M</td></tr>
<tr><td>Label-String</td><td>See IEEE Std 11073-20601a-2010.</td><td>O</td></tr>
<tr><td>Unit-LabelString</td><td>See IEEE Std 11073-20601a-2010.</td><td>O</td></tr>
<tr><td>Absolute-Time-Stamp</td><td>See 6.7.1.3.</td><td>C</td></tr>
</table>
<table>
<tr><td>Nu-Observed-Value</td><td>See IEEE Std 11073-20601a-2010.</td><td>C</td></tr>
</table>
<table>
<tr><td>Unit-Code</td><td>Valid values are limited to MDC_DIM_CAL or MDC_DIM_JOULES.</td><td>M</td></tr>
</table>
<table>
<tr><td>HF | Energy | Expended</td><td>Energy expended</td><td>The number of calories expended over a period of time.</td><td>MDC_ HF_ENERGY</td><td>119</td></tr>
</table>
</td><td>1.0</td></tr>
<tr><td>HR-HSF-03.1</td><td>If the device supports energy expended measurements, the Heart Rate Plug-In SHALL report the value of the TYPE attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit partition and 16 bit code; partition:code). <i>[The TYPE code attribute value for the Energy Expended object is fixed]</i>
Example:
String: “energy expended”
Code: “119”</td><td>1.0</td></tr>
</table>

Label	Description	Release
HR-HSF-03.2	If the device supports energy expended measurements, the Heart Rate Plug-In SHALL report the value reported from the appropriate *-Nu-Observed-Val attribute in two formats, i.e., float and string: 1- Float that represents the Heart Rate report data measured by the targeted Heart Rate monitor and represented in a float format. This is for general use of application programming. 2- String that represents the Heart Rate report data measured by the targeted Heart Rate monitor and reported to the Plug-In in an MDER FLOAT format. Since there is no way to transfer an MDER FLOAT from Plug-Ins to applications, an MDER FLOAT is represented as a hexadecimal string, such as "FFFFC8E". In this way, the MDER FLOAT can preserve the exact precision as derived from the MDER encoded FLOAT. (Note that IEEE devices will report this value typically as an MDER SFLOAT which the Temperature Plug-In maps to an MDER FLOAT.) MDER FLOAT may be sent to the cloud for other applications while preserving the original precision provided by the MDER FLOAT. Example: String: “5” MDER FLOAT: “00000005”	1.0
HR-HSF-03.3	If the device supports energy expended measurements, the Heart Rate Plug-In SHALL report the value of the Unit Code attribute as a human readable string and as its 32-bit MDC code (combine the 16-bit bit partition code which is always 4 with the 16-bit code partition:code or 4:code). Example: String: “Calories” Code: “6784”	1.0
HR-HSF-03.4	If the device supports energy expended measurements, the Heart Rate Plug-In SHALL report the measurement time stamp as an HL7 DTM time stamp as specified by HD-HLF-06 and HD-HLF-07. Example: Timestamp: 20150506164232.00-0400	1.0

Editor Note: Heart rate

Type as String: “Heart rate” Type as Code: “147842”	
Type as String: “75” MDER FLOAT: “0000002D”	Units as string: “beats per min” Units as code “264864”
Timestamp: “20150504135813.22-0400”	

Editor Note: Energy Expended

Type as String: “energy expended” Type as Code: “119”	
Type as String: “5” MDER FLOAT: “00000005”	Units as string: “Calories ” Units as code: “6784”
Timestamp: “20150504135813.22-0400”	

6.8.1.1 Data Integrity: DWAPI-PCH

6.8.1.2 Confidentiality

7. Architectural Model

This section describes the architectural model and related aspects of the Device WebAPI Enabler.

7.1 Architectural Diagram

7.1.1 GotAPI Framework Summary

This section summarizes how the GotAPI framework works, in which the DWAPI is functioning. This section adheres to the specifications that are defined by GotAPI 1.1 [GotAPI 1.1].

As defined by GotAPI 1.1 [GotAPI 1.1], when an application is initiated by a user, the application obtains authorization for access to GotAPI-based APIs using the GotAPI-2 Interface. Once the application is authorized by the GotAPI Auth Server, the application can access the GotAPI Server using the GotAPI-1 Interface.

After the authorization, the application asks the GotAPI Server, using the GotAPI-1 Interface, what kind of services are available. Then the GotAPI Server requests the current status of the available services to all the installed Extension Plug-Ins using the GotAPI-4 Interface. This procedure is called the "Service Discovery", which is defined in the GotAPI specification. After the Service Discovery, the application can interact with the specified device (i.e., the service) via the Plug-In. Note that, in the GotAPI specification, external devices and internal enablers are collectively called as “services”.

When an application sends an API request on the GotAPI-1 Interface, the GotAPI Server passes it to the Plug-In using the GotAPI-4 Interface.

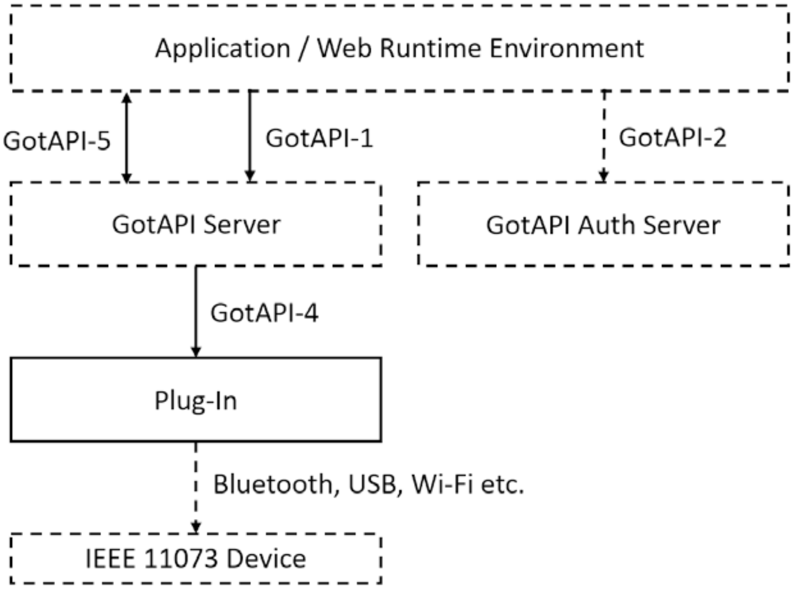


Figure 3: Architectural Diagram

In addition to HTTP, an application may connect to the GotAPI Server using WebSocket, which is the GotAPI-5 Interface. The GotAPI-5 Interface enables that whenever the targeted device reports event messages, the application receives the messages on the GotAPI-5 Interface asynchronously.

The GotAPI Server is agnostic to what Plug-Ins do inside. The GotAPI Server just passes a request from an application to a Plug-In and passes a response from the Plug-In to the application.

7.1.2 GotAPI Framework and IEEE 11073 Healthcare Devices

This section describes how the GotAPI framework and IEEE 11073 healthcare devices work together using Extension Plug-Ins. The following diagram shows the basic flow of DWAPI-PCH.

Under the GotAPI framework, the Plug-In implements web-based APIs, DWAPI-PCH, and the Manager whose function is defined by IEEE 11073. The Plug-In with the Manager communicates with IEEE 11073 healthcare devices that implements

Agent and Sensor through some media such as Bluetooth or WIFI. IEEE 11073 defines all the necessary protocols, data formats, and the roles for Manager and Agent. From the Manager, some data is made available to DWAPI-PCH to be exposed in the web-based APIs to applications through the GotAPI framework. The Plug-In makes such data available to applications through DWAPI-PCH consistently under the GotAPI framework.

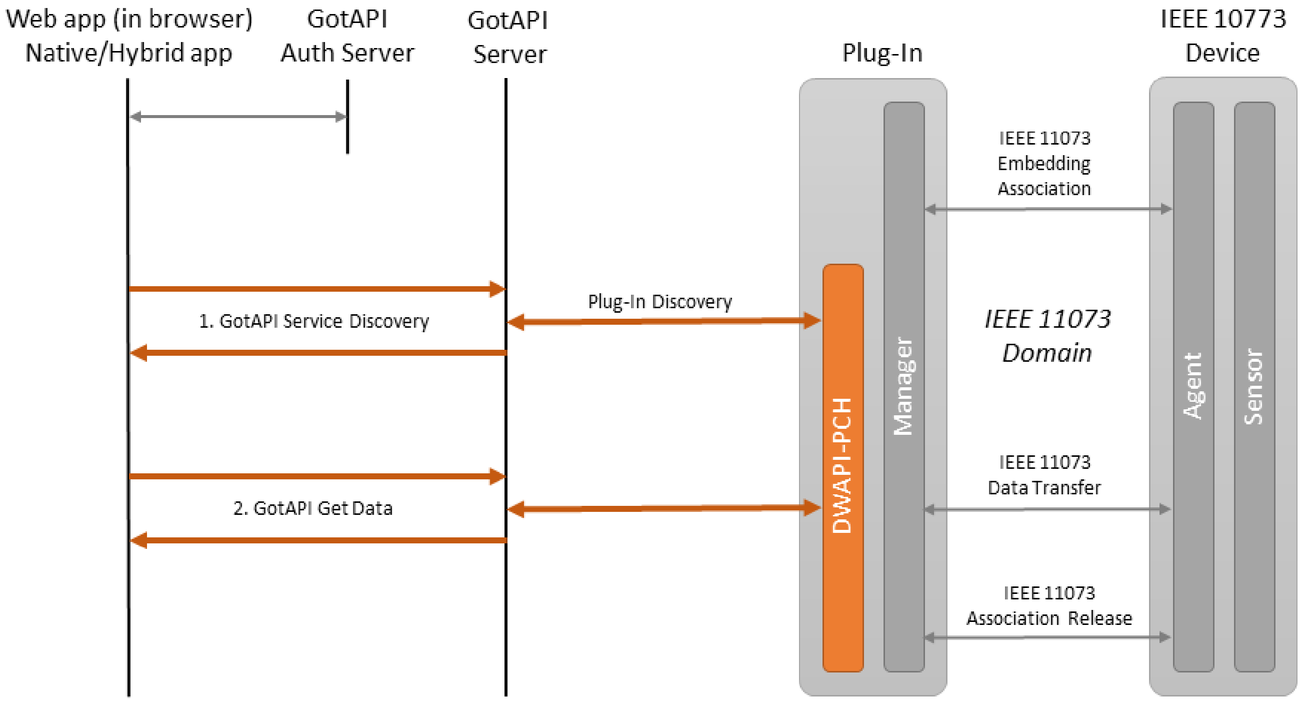


Figure 4: DWAPI-PCH Basic data flows

7.2 Functional Components and Interfaces/reference points definition

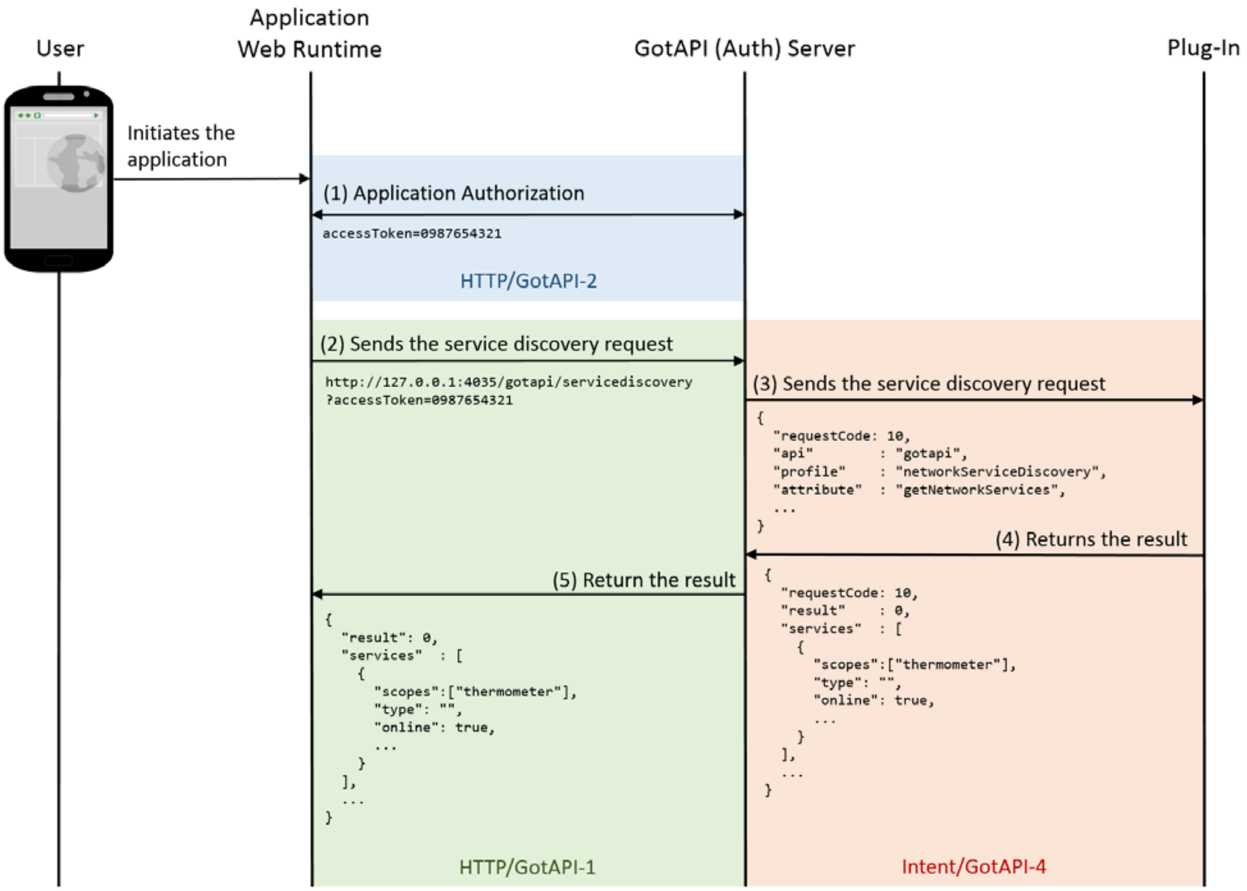
DWAPI-PCH consists of the following three APIs;

1) The Service Discovery API enables applications to obtain information of Plug-Ins and IEEE 11073 devices available.
2) The One-shot measuring API enables applications to get one set of measuring values in response to a request.
3) The asynchronous messaging API enables applications to listen to asynchronous messages from the targeted device via the relevant Plug-In.

7.2.1 Service Discovery API

Service Discovery API specification adheres to that of GotAPI 1.1.

As defined by GotAPI 1.1, after the application obtains authorization for access to GotAPI-based APIs using the GotAPI-2 Interface, the application sends the Service Discovery request to the GotAPI Server. Then the GotAPI Server sends the Service Discovery request to all of the installed Extension Plug-Ins. The message flow of the Service Discovery is shown in Fig. 5.



```
accessToken=0987654321
```

```
http://127.0.0.1:4035/gotapi/servicediscovery
?accessToken=0987654321
```

```
{
  "requestCode: 10,
  "api"        : "gotapi",
  "profile"    : "networkServiceDiscovery",
  "attribute"  : "getNetworkServices",
  ...
}
```

```
{
  "requestCode: 10,
  "result"     : 0,
  "services"   : [
    {
      "scopes":["thermometer"],
      "type": "",
      "online": true,
      ...
    }
  ],
  ...
}
```

```
{
  "result": 0,
  "services"  : [
    {
      "scopes":["thermometer"],
      "type": "",
      "online": true,
      ...
    }
  ],
  ...
}
```

Figure 5: Message flow of the Service Discovery: Message flow of the Service Discovery

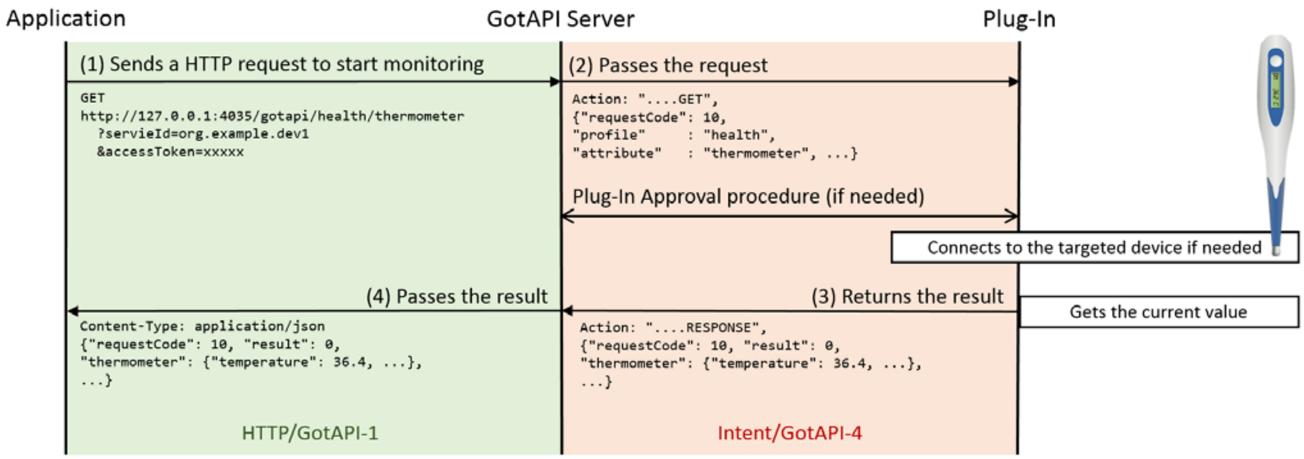
The specific data in the message flows labelled (4) in the figure above are defined by the Plug-In that implements Manager functionality of IEEE 11073 and is communicating with healthcare devices.

The other message flows SHALL be consistent to what are defined in the GotAPI 1.1 specification.

7.2.2 One-shot measuring API

As defined by GotAPI 1.1, after the application obtains authorization to access GotAPI-based APIs using the GotAPI-2 Interface and completes the Service Discovery, the application can use the service (so called "One-shot measuring API") provided by the Plug-In through the GotAPI Server.

The One-shot measuring API offers a measurement result reported by the targeted device in response to a request. The message flow of this API is as shown blow.



```
GET
http://127.0.0.1:4035/gotapi/health/thermometer
  ?servieId=org.example.dev1
  &accessToken=xxxxx
```

```
Action: "....GET",
{"requestCode": 10,
"profile"     : "health",
"attribute"   : "thermometer", ...}
```

```
Content-Type: application/json
{"requestCode": 10, "result": 0,
"thermometer": {"temperature": 36.4, ...},
...}
```

```
Action: "....RESPONSE",
{"requestCode": 10, "result": 0,
"thermometer": {"temperature": 36.4, ...},
...}
```

Figure 6: Message flow of the One-shot measuring API

1. The user triggers a request of the API in the application.
2. Label (1): The application sends a request to the GotAPI Server using HTTP (REST) over the GotAPI-1 Interface. Note that the HTTP method of the request is "GET".
3. Label (2): The GotAPI Server passes the request to the targeted Plug-In on the GotAPI-4 Interface with the Action name "GET".
4. The GotAPI Server runs the Plug-In Approval procedure if needed, which is defined in the GotAPI 1.1 specification.
5. When the Plug-In receives the request, it connects to the targeted external device if needed.
6. The Plug-In obtains current measurement values from the targeted device.
7. Label (3): The Plug-In sends a response with one set of the measurement values using the GotAPI-4 Interface.
8. La*bel (4)*: When the GotAPI Server receives the response from the Plug-In, the GotAPI Server passes the response to the application on the GotAPI-1 Interface as an HTTP response.

The overall message flows to obtain data by sending HTTP request and response over the GotAPI-1 Interface SHALL adhere to the specifications defined in GotAPI 1.1.

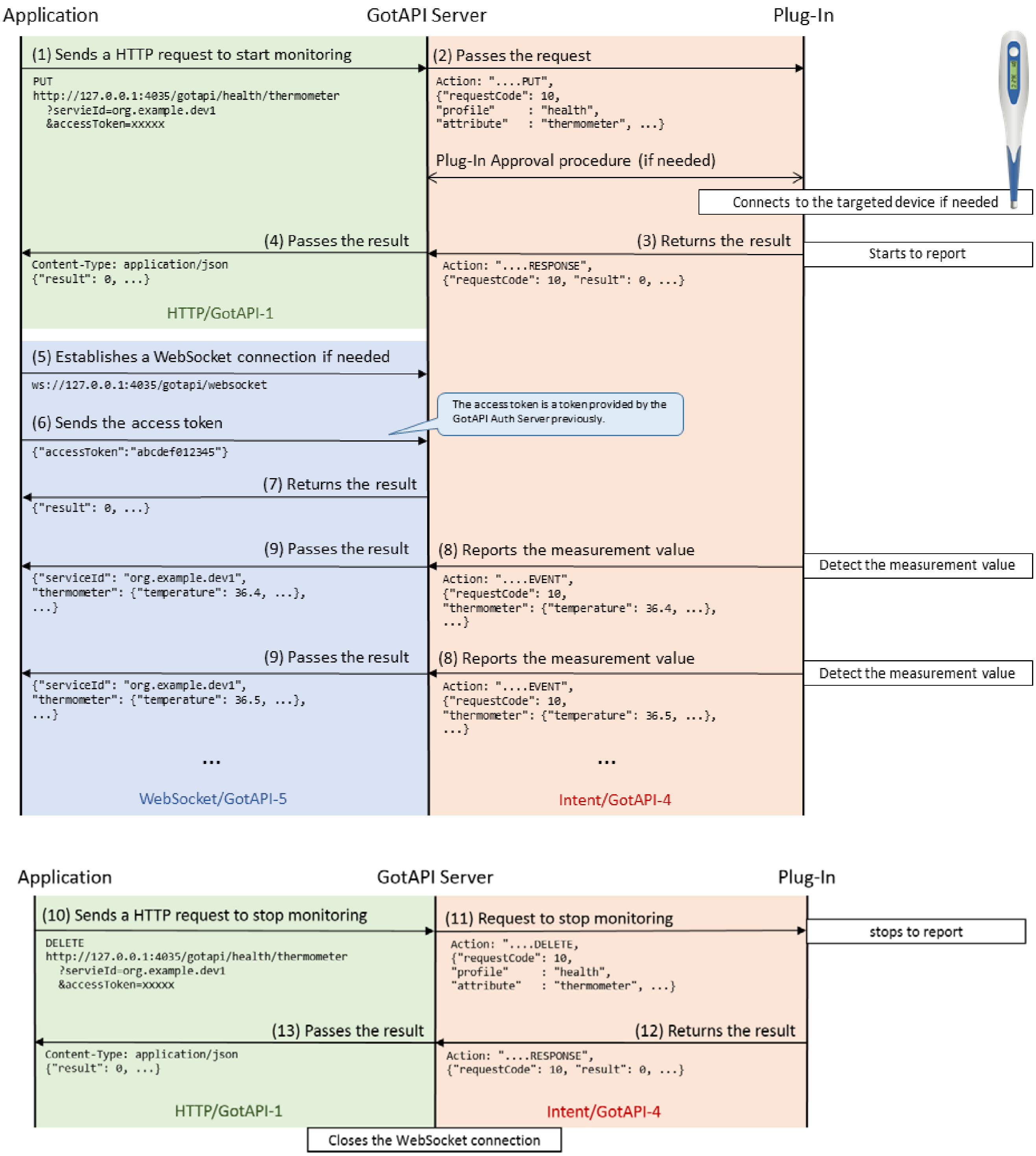
The specific data in the message flows labelled (3) and (4) in the figure above are defined by the Plug-In that implements Manager functionality of IEEE 11073 and is communicating with healthcare devices.

7.2.3 Asynchronous messaging API

As defined by GotAPI 1.1, after the application obtains authorization to access GotAPI-based APIs using the GotAPI-2 Interface and completes the Service Discovery, the application can use the service (so called "Asynchronous messaging API") provided by the Plug-In through the GotAPI Server.

The Asynchronous messaging API offers a series of measurement values reported by the targeted device to an application in real time as the measurement values become available. The timing when and the reasons why such measurement values become available is determined by the Plug-Ins and connected devices, and is out of the scope of this specification.

This API uses WebSocket protocol to handle asynchronous event messages. The message flow of this API is shown blow:


Application
GotAPI Server
Plug-In
(1) Sends a HTTP request to start monitoring
(2) Passes the request
Plug-In Approval procedure (if needed)
Connects to the targeted device if needed
(4) Passes the result
(3) Returns the result
Starts to report
HTTP/GotAPI-1
(5) Establishes a WebSocket connection if needed
The access token is a token provided by the GotAPI Auth Server previously.
(6) Sends the access token
(7) Returns the result
(9) Passes the result
(8) Reports the measurement value
Detect the measurement value
(9) Passes the result
(8) Reports the measurement value
Detect the measurement value
…
…
WebSocket/GotAPI-5
Intent/GotAPI-4
Application
GotAPI Server
Plug-In
(10) Sends a HTTP request to stop monitoring
(11) Request to stop monitoring
stops to report
(13) Passes the result
(12) Returns the result
HTTP/GotAPI-1
Intent/GotAPI-4
Closes the WebSocket connection

```
PUT
http://127.0.0.1:4035/gotapi/health/thermometer
  ?servieId=org.example.dev1
  &accessToken=xxxxx
```

```
Action: "....PUT",
{"requestCode": 10,
"profile"     : "health",
"attribute"   : "thermometer", ...}
```

```
Action: "....RESPONSE",
{"requestCode": 10, "result": 0, ...}
```

```
Content-Type: application/json
{"result": 0, ...}
```

```
ws://127.0.0.1:4035/gotapi/websocket
```

```
{"accessToken":"abcdef012345"}
```

```
{"result": 0, ...}
```

```
Action: "....EVENT",
{"requestCode": 10,
"thermometer": {"temperature": 36.4, ...},
...}
```

```
{"serviceId": "org.example.dev1",
"thermometer": {"temperature": 36.4, ...},
...}
```

```
Action: "....EVENT",
{"requestCode": 10,
"thermometer": {"temperature": 36.5, ...},
...}
```

```
{"serviceId": "org.example.dev1",
"thermometer": {"temperature": 36.5, ...},
...}
```

```
DELETE
http://127.0.0.1:4035/gotapi/health/thermometer
  ?servieId=org.example.dev1
  &accessToken=xxxxx
```

```
Action: "....DELETE,
{"requestCode": 10,
"profile"     : "health",
"attribute"   : "thermometer", ...}
```

```
Action: "....RESPONSE",
{"requestCode": 10, "result": 0, ...}
```

```
Content-Type: application/json
{"result": 0, ...}
```

Figure 7: Message Flow of the Asynchronous messaging API

1. The user triggers a request of the API in the application.
2. Label (1): The application sends a request to the GotAPI Server using HTTP (REST) over the GotAPI-1 Interface. Note that the HTTP method of the request is "PUT".

3. Label (2): The GotAPI Server passes the request to the targeted Plug-In on the GotAPI-4 Interface with the Action name "PUT".
4. The GotAPI Server runs the Plug-In Approval procedure if needed, which is defined in the GotAPI 1.1 specification.
5. When the Plug-In receives the request, it connects to the targeted external device if needed.
6. Label (3): The Plug-In sends a response with the message using the GotAPI-4 Interface.
7. Label (4): When the GotAPI Server receives the response from the Plug-In, the GotAPI Server passes the response to the application on the HTTP connection as an HTTP response.
8. Label (5): The application establishes a WebSocket connection to the GotAPI Server if the application does not have a WebSocket connection to the GotAPI Server yet.
9. Label (6): As the WebSocket connection has been established, the application sends the access token to the GotAPI Server through the WebSocket connection. The access token is a token which the application obtained from the GotAPI Auth Server when the application was authorized by the GotAPI Auth Server.
10. Label (7): When the GotAPI Server receives the access token from the WebSocket channel, the GotAPI Server returns the result on whether the request is accepted or not.
11. Label (8): Whenever the targeted external device reports a message, e.g., a data or a measurement value, the Plug-In sends the message to the GotAPI Server on the GotAPI-4 Interface with the Action name "EVENT".
12. Label (9): Whenever the GotAPI Server receives a message from the Plug-In, the GotAPI Server passes it to the application on the WebSocket connection.
13. Label (10): When the application finishes or decides to finish using the service, it sends a request to stop the monitoring to the GotAPI Server. The request is sent over the GotAPI-1 Interface using HTTP. Note that the URI is the same as that of the first request except that the HTTP method is "DELETE".
14. Label (11): When the GotAPI Server receives the stop request, it sends a request to the Plug-In to stop the monitoring with the Action name "DELETE". Then the GotAPI server closes the WebSocket connection.
15. Label (12): When the Plug-In receives the stop request from the GotAPI Server, the Plug-In stops reporting messages, and it returns a response to the GotAPI Server on the GotAPI-4 Interface with the Action name “RESPONSE”.
16. Label (13): When the GotAPI Server receives the response, the GotAPI Server passes the response to the application on the GotAPI-1 Interface.

The diagram above shows that the application establishes a WebSocket connection as the GotAPI-5 Interface after the application sends an API request on the GotAPI-1 Interface. It should be noted, as defined in GotAPI 1.1, the application is permitted to establish a WebSocket connection only after the application has received an access token from the GotAPI Auth Server.

The overall message flows to establish/close an asynchronous messaging session and to receive measurement values asynchronously from Plug-Ins SHALL adhere to the specifications defined in GotAPI 1.1.

The specific data in the message flows labelled (1) to (13) in the figure above are defined by the Plug-In that implements Manager Functionality of IEEE 11073 and is communicating with healthcare devices.

7.3 Behaviors of Plug-Ins for reporting measurements to applications

7.3.1 Measurement modes and one shot/asynchronous messaging

There are two measurement modes for personal connected healthcare (PCH) devices, (i) the single measurement mode and (ii) the continuous measurement mode. Depending on the nature of measurements that the device generates, some support only one mode while others support both modes.

The single measurement mode provides only one measurement per one measurement process, whereas the continuous measurement mode provides multiple measurements continuously. An example of the single measurement is weight scales, which provide one measurement of weight shortly after a user steps onto the scale. An example of continuous measurement mode is heart rate monitors, which provide a measurement of user’s heart rate periodically, say every one second, and continuously.

Table 4 shows measurement mode support for various devices.

	Measurement mode	
	Single	Continuous
Thermometer	Yes	No
Glucometer	Yes	No
Weight Scale	Yes	No
BCA	Yes	No
Blood Pressure	Yes	No
Heart Rate	No	Yes
Pulse Oximeter	Yes	Yes

Table 4 Measurement modes for various devices

Generally, the implementation of PCH devices measure and provide the data in the following steps:

1. The user completes pairing of a PCH device and a smartphone. Pairing must be completed before any measurements or data transfer to take place.
2. The user starts up the application on the smartphone, and takes a measurement using the PCH device.

3A. Single measurement mode:

When the PCH device acquires a measurement result successfully;

(i) the connection between the PCH device and the smartphone (i.e., the Plug-In) is established,

(ii) the measurement result is sent to the smartphone, and

(iii) the connection is closed by the PCH device.

3B. Continuous measurement mode:

When the PCH device acquires a measurement result successfully;

(i) the connection between the PCH device and the smartphone (i.e., the Plug-In) is established,

(ii) the measurement result is sent to the smartphone,

(iii) the series of results that are continuously measured afterward are sent to the smartphone one after another whenever they are acquired by the PCH device, e.g., every one second, and

(iv) the connection is closed by the PCH device, e.g., the user stopped measurement with the PCH device or an error occurred with the measurement.

The Plug-In is not able to detect the status of measurement being conducted by the PCH device. The Plug-In just receives measurement results only when the measurement succeeds.

7.3.2 Policy for one-shot messages

One-shot messages can be sent to a Plug-In by an application arbitrarily at any time without knowing the status of the Plug-In or the measurement that is underway. Therefore, in order to clarify what applications can obtain by use of one-shot messages, the policy for responding to a one-shot API request is specified as follows:

1. If the Plug-In has the latest measurement result, the Plug-In SHALL return the result immediately.
2. Otherwise, the Plug-In SHALL return an error.

This means that if the application receives an error, it may have to send additional one-shot API requests to the Plug-In in order to obtain measurement data for single measurement devices. For continuous measurement devices, the application need to send one-shot API requests one after another in order to get new measurement data that are sent from the PCH device.

The following figures show examples of responses to one-shot API requests from Plug-Ins depending on various measurement status.

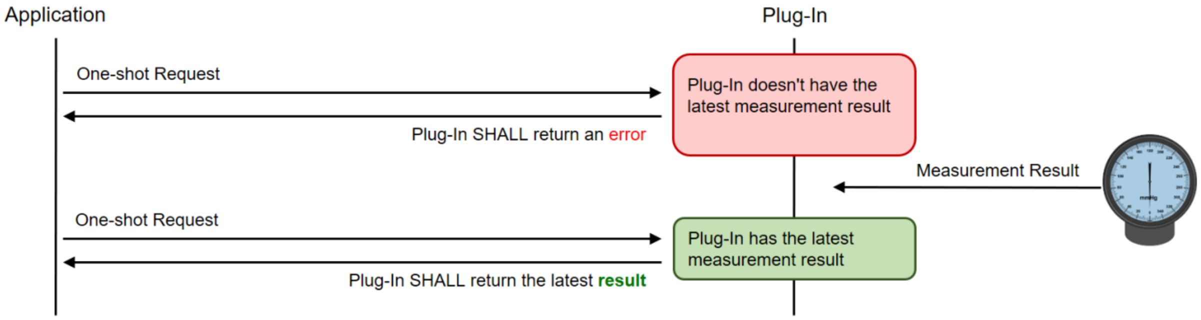

Application
Plug-In
One-shot Request
Plug-In doesn't have the latest measurement result
Plug-In SHALL return an error
Measurement Result
One-shot Request
Plug-In has the latest measurement result
Plug-In SHALL return the latest result

Figure 8: Example of single measurement.

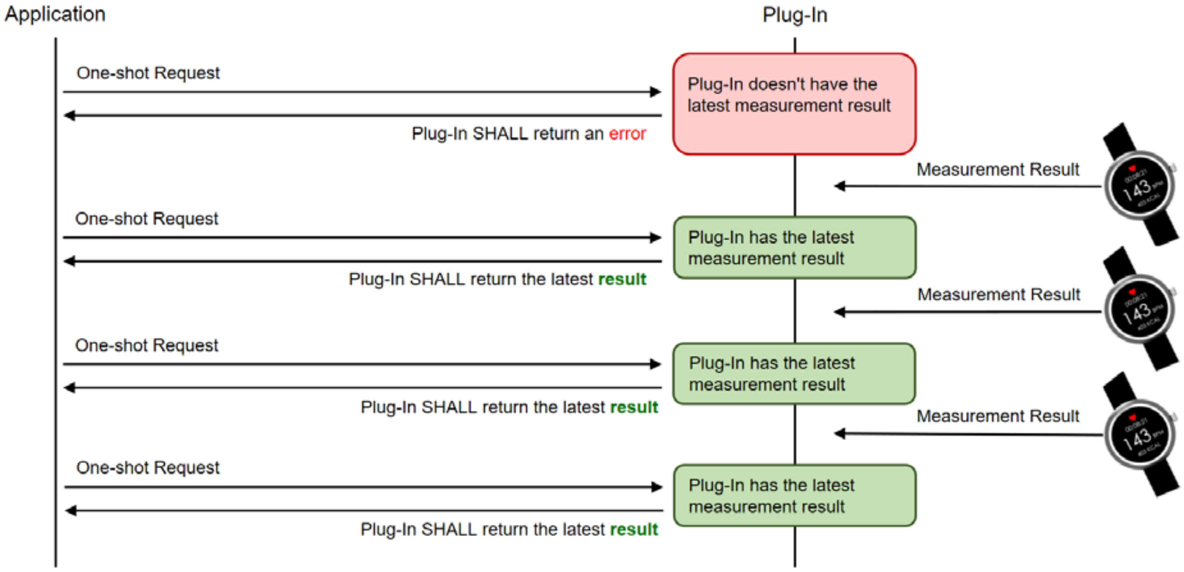

Application
Plug-In
One-shot Request
Plug-In doesn't have the latest measurement result
Plug-In SHALL return an error
Measurement Result
One-shot Request
Plug-In has the latest measurement result
Plug-In SHALL return the latest result
Measurement Result
One-shot Request
Plug-In has the latest measurement result
Plug-In SHALL return the latest result
Measurement Result
One-shot Request
Plug-In has the latest measurement result
Plug-In SHALL return the latest result

Figure 9: Example of continuous measurement.

7.3.3 Policy for asynchronous messages

When the Plug-In receives a request of asynchronous messaging API over the GotAPI-1 Interface:

- If the Plug-In has the latest result, it SHALL return the result to the application immediately over the GotAPI-5 Interface.
- Otherwise, the Plug-In SHALL NOT send anything.

After the Plug-In receives a request of asynchronous messaging API over the GotAPI-1 Interface:

- Whenever the Plug-In gets the latest result from the connected PCH device, it SHALL return the result to the application immediately over the GotAPI-5 Interface.
- At other times, the Plug-In SHALL NOT send anything.

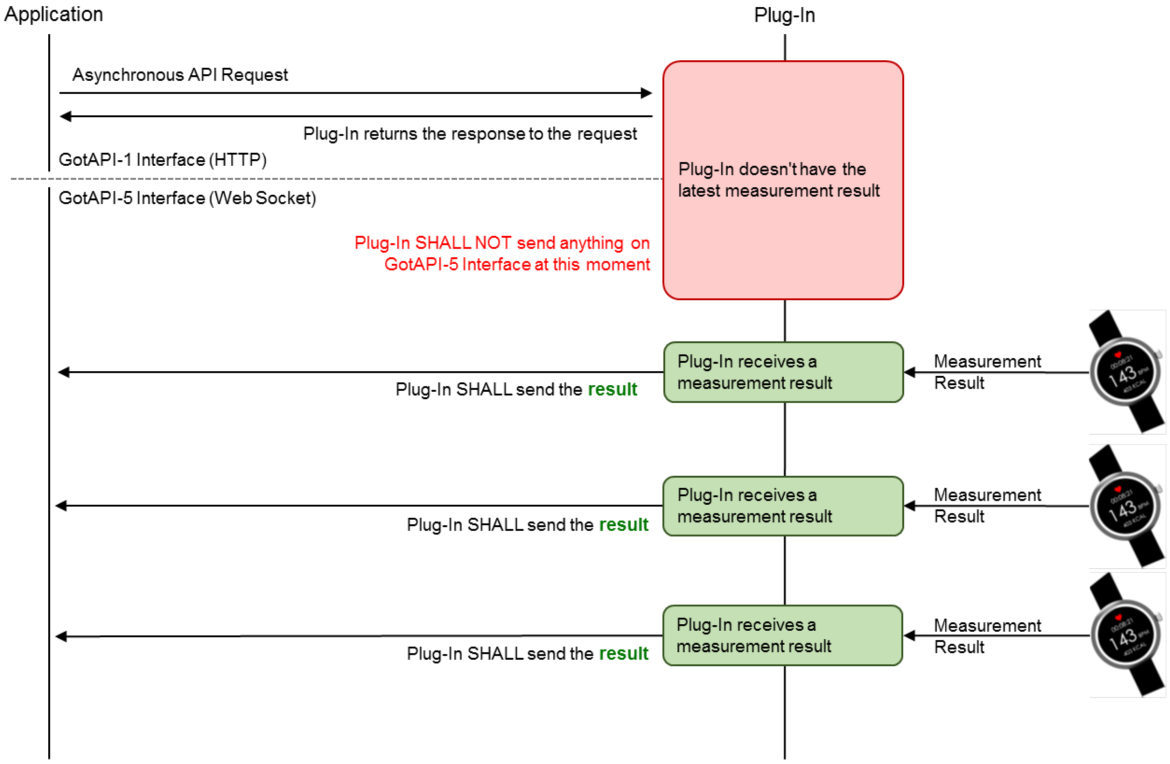

Application
Plug-In
Asynchronous API Request
Plug-In returns the response to the request
GotAPI-1 Interface (HTTP)
GotAPI-5 Interface (Web Socket)
Plug-In doesn't have the latest measurement result
Plug-In SHALL NOT send anything on GotAPI-5 Interface at this moment
Plug-In receives a measurement result
Measurement Result
Plug-In SHALL send the result
Plug-In receives a measurement result
Measurement Result
Plug-In SHALL send the result
Plug-In receives a measurement result
Measurement Result
Plug-In SHALL send the result

Figure 10: Example of asynchronous messages (continuous measurement).

7.3.4 Intermediate measurements

Some personal healthcare devices may provide intermediate measurements in addition to actual measurements (“measurements”). For example, intermediate temperature for thermometers, and intermediate cuff pressure for blood pressure monitors. Intermediate measurements are provided for display purposes while the measurement process is in progress. Intermediate measurements are supported as optional features in the IEEE and Bluetooth LE (BLE) specifications.

However, currently there are no personal connected healthcare products available in the market, at which this API is targeting, supporting intermediate measurements. Therefore the current version of this specification does not standardize intermediate measurement features.

- The Plug-Ins SHALL not send an intermediate measurement as a measurement to applications.

If a Plug-In wants to send intermediate measurements to applications, it needs to define a proprietary data set of its own for intermediate measurements so as not to be confused with a measurement by applications. For example, use a unique error code that is not used elsewhere to indicate that the value is an intermediate measurement. The details of the error codes are defined in the Technical Specification sections of each PCH devices.

7.4 Security Considerations

This specification SHALL adhere to all the security requirements that are defined in GotAPI 1.1.

The GotAPI 1.1 specification considers every security risks and implements necessary counter measures for them. For example:

- The GotAPI 1.1 has an application-authorization mechanism. Applications can't access the APIs without user permissions. Besides, when applications access devices via Plug-Ins, the relevant Plug-In obtains a permission from the user.

- The GotAPI 1.1 has an HMAC server authentication mechanism. Applications are able to detect if the GotAPI Server is spoofed.

See the section "7.3 Security Considerations" in the GotAPI 1.1 specification for the details of the security considerations of the GotAPI 1.1.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
n/a	n/a	No prior version

A.2 Draft/Candidate Version 1.0 History

Document Identifier	Date	Sections	Description
Draft Versions OMA-ER-Device_WebAPIs-V1_0	25 Jan 2015	All	Initial baseline document.
	22 May 2015	6	Added sections 6.1, 6.2 and 6.3
	13 Jun 2015	6	Added sections 6.4. 6.5.6.6, 6.7 and updated sectons 6.2 and 6.3
	21 Aug 2015	All	Added section 7, 6.1, update section 4.
	28 Dec 2015	Add	CR-24, 25, 26, 27, 46, 47, 48R03
	23 Jan 2016	6.4, 6.5, 6.6. 6.7. 6.8	CR 08, CR 10, CR 11, CR 12 and CR 13 for 2016
Candidate Version OMA-ER-Device_WebAPIs-V1_0	19 Apr 2016	All	Status changed to Candidate by TP TP Ref # OMA-TP-2016-0057-INP_DWAPI_V1_0_ERP_for_Candidate_approval

Appendix B. Call Flows (Informative)

This is a placeholder to be populated, as required.

Appendix C. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

C.1 ERDEF for Device WebAPI 1.0 - Client Requirements

This section is normative.

Item	Feature / Application	Requirement
OMA-ERDEF-Device_WebAPIs_1.0-C-001-<<M/O>>		

Table 5: ERDEF

C.2 ERDEF for GotAPI 1.0 - Server Requirements

This section is normative.

Item	Feature / Application	Requirement
OMA-ERDEF-GotAPI_1.0-S-001-<<M/O>>	GotAPI 1.0 Server	

Table 6: ERDEF for GotAPI 1.0 Server-side Requirements

Appendix D. Device WebAPI Enabler Deployment Considerations

This is a placeholder, to be populated as required.

Appendix E. List of IEEE 11073 specifications

Editor's Note: The status may need to be updates.

Status	Specification	Title
Completed Standards	IEEE Std 11073-10404 Dev specialization	Pulse oximeter
Completed Standards	IEEE Std 11073-10406 Dev specialization	Basic ECG
Completed Standards	IEEE Std 11073-10407 Dev specialization	Blood pressure monitor
Completed Standards	IEEE Std 11073-10408 Dev specialization	Thermometer
Completed Standards	IEEE Std 11073-10415 Dev specialization	Weighing scale
Completed Standards	IEEE Std 11073-10417 Dev specialization	Glucose meter + Revision
Completed Standards	IEEE Std 11073-10418 Dev specialization	INR (blood coagulation)
Completed Standards	IEEE Std 11073-10420 Dev specialization	Body composition analyzer
Completed Standards	IEEE Std 11073-10421 Dev specialization	Peak flow
Completed Standards	IEEE Std 11073-10441 Dev specialization	Cardiovascular + Revision
Completed Standards	IEEE Std 11073-10442 Dev specialization	Strength
Completed Standards	IEEE Std 11073-10471 Dev specialization	Activity hub
Completed Standards	IEEE Std 11073-10472 Dev specialization	Medication monitor
Completed Standards	IEEE Std 11073-20601	Optimized exchange protocol + Amendment
Completed Standards	IEEE Std 11073-00103	Guide for Health informatics - Personal health device communication - Overview
Work being drafted	IEEE Std 11073-20601	Optimized exchange protocol (Revision)
Work being drafted	IEEE P11073-10404 Dev specialization	Pulse oximeter (Revision)
Work being drafted	IEEE P11073-10413 Dev specialization	Respiration rate
Work being drafted	IEEE P11073-10419 Dev specialization	Insulin pump
Work being drafted	IEEE P11073-10422 Dev specialization	Urine analyzer
Work being drafted	IEEE P11073-10423 Dev specialization	Sleep Quality Monitor

Work being drafted	IEEE P11073-10424 Dev specialization	Sleep Aponea Breathing Therapy Equipment
Work being drafted	IEEE P11073-10425 Dev specialization	Continuous Glucose Meter
Work being drafted	IEEE P11073-10417a Dev specialization	Glucose meter (Amendment)
Work being drafted	IEEE P11073-10406a Dev specialization	Basic ECG (Amendment)
Work being drafted	IEEE P11073-10471a Dev specialization	AI Living Hub (Amendment)
Work being drafted	IEEE P11073-10407 Dev specialization	Blood Pressure Monitor (Corrigendum)
Work being drafted	IEEE P11073-10408 Dev specialization	Thermometer (Corrigendum)
Work being drafted	IEEE P11073-10415 Dev specialization	Weighing Scale (Corrigendum)
Work being drafted	IEEE P11073-10420 Dev specialization	Body composition analyzer (Corrigendum)
Work being drafted	IEEE P11073-10418 Dev specialization	INR monitor (Corrigendum)