Generic Health Sensor Service

***Bluetooth*® Service Specification**

- **Version:** v1.0
- **Version Date:** 2023-06-13
- **Prepared By:** Medical Devices Working Group

Abstract:

This specification defines the server component of the Generic Health Sensor Profile and Service specifications (GHS). GHS defines an exchange protocol that can be used to communicate health-related observations from different types of personal health devices. The generic nature of GHS is achieved by modeling devices and their observations through the composition of simple components. Components contain self-describing metadata where clinically related concepts are expressed using standardized nomenclature terms.

Version History

Version Number	Date (yyyy-mm-dd)	Comments
v1.0	2023-06-13	Adopted by the Bluetooth SIG Board of Directors.

Acknowledgments

Name	Company
Erik Moll	Koninklijke Philips N.V.
Barry Reinhold	Lamprey Networks
Brian Reinhold	Lamprey Networks
Abdul Nabi	Koninklijke Philips N.V.

Contents

1 Introduction

The Generic Health Sensor (GHS) specifications define a protocol by which personal health device data can be exchanged over a Bluetooth connection. GHS is composed of the Generic Heath Sensor Service (GHSS) and the Generic Health Sensor Profile (GHSP) specifications. This specification documents the GHSS. GHSS communicates the semantic content contained in the Abstract Content Information Model (ACOM) defined in IEEE 11073-10206 [6] using the Generic Attribute Protocol (GATT). GHSS, in conjunction with the ACOM, provides a unified, extensible approach to personal health device communications and can be used in place of previous services and profiles written for specific health sensor device types. GHSS is also applicable to other types of personal health devices as well as proprietary sensor devices that report observations in a standardized health infrastructure.

More background information can be found in the introduction of GHSP [7].

1.1 Language

1.1.1 Language conventions

In the development of a specification, the Bluetooth SIG has established the following conventions for use of the terms *"**shall**"*, *"shall not"*, *"**should**"*, *"should not"*, *"**may**"*, *"**must**"*, and *"**can**".* In this Bluetooth specification, the terms in Table 1.1 have the specific meanings given in that table, irrespective of other meanings that exist.

Term	Definition
shall	—used to express what is required by the specification and is to be implemented exactly as written without deviation
shall not	—used to express what is forbidden by the specification
should	—used to express what is recommended by the specification without forbidding anything
should not	—used to indicate that something is discouraged but not forbidden by the specification
may	—used to indicate something that is permissible within the limits of the specification
must	—used to indicate either: 1. an indisputable statement of fact that is always true regardless of the circumstances 2. an implication or natural consequence if a separately-stated requirement is followed
can	—used to express a statement of possibility or capability

Table 1.1: Language conventions terms and definitions

1.1.1.1 Implementation alternatives

When specification content indicates that there are multiple alternatives to satisfy specification requirements, if one alternative is explained or illustrated in an example it is not intended to limit other alternatives that the specification requirements permit.

1.1.1.2 Discrepancies

It is the goal of Bluetooth SIG that specifications are clear, unambiguous, and do not contain discrepancies. However, members can report any perceived discrepancy by filing an erratum and can request a test case waiver as appropriate.

1.1.2 Reserved for Future Use

Where a field in a packet, Protocol Data Unit (PDU), or other data structure is described as "Reserved for Future Use" (irrespective of whether in uppercase or lowercase), the device creating the structure shall set its value to zero unless otherwise specified. Any device receiving or interpreting the structure shall ignore that field; in particular, it shall not reject the structure because of the value of the field.

Where a field, parameter, or other variable object can take a range of values, and some values are described as "Reserved for Future Use," a device sending the object shall not set the object to those values. A device receiving an object with such a value should reject it, and any data structure containing it, as being erroneous; however, this does not apply in a context where the object is described as being ignored or it is specified to ignore unrecognized values.

When a field value is a bit field, unassigned bits can be marked as Reserved for Future Use and shall be set to 0. Implementations that receive a message that contains a Reserved for Future Use bit that is set to 1 shall process the message as if that bit was set to 0, except where specified otherwise.

The acronym RFU is equivalent to Reserved for Future Use.

1.1.3 Prohibited

When a field value is an enumeration, unassigned values can be marked as “Prohibited.” These values shall never be used by an implementation, and any message received that includes a Prohibited value shall be ignored and shall not be processed and shall not be responded to.

Where a field, parameter, or other variable object can take a range of values, and some values are described as “Prohibited,” devices shall not set the object to any of those Prohibited values. A device receiving an object with such a value should reject it, and any data structure containing it, as being erroneous.

“Prohibited” is never abbreviated.

1.2 Table requirements

Requirements in tables in this document are defined as "Mandatory" (M), "Optional" (O), "Excluded" (X), “Not Applicable” (N/A), or "Conditional" (C.n). Conditional statements (C.n) are listed directly below the table in which they appear.

1.3 Conformance

Each capability of this specification shall be supported in the specified manner. This specification may provide options for design flexibility, because, for example, some products do not implement every portion of the specification. For each implementation option that is supported, it shall be supported as specified.

1.4 Byte transmission order

All characteristics used with this service shall be transmitted with the least significant octet (LSO) first (i.e., little endian). Where the format is described in tables in this document, the LSO is the first octet in the topmost field of the table; the most significant octet (MSO) is the last octet in the bottommost field of the table. Where characteristics are defined in the GATT Specification Supplement (GSS), see Section 2.2 in GSS [1].

1.5 IEEE FLOAT

This service defines several characteristics with fields containing a FLOAT data type. This data type is the IEEE 11073 32-bit FLOAT data type as defined in [9]. A FLOAT value consists of 4 octets, starting with a one octet decimal exponent followed by a 3-octet mantissa. Both the exponent and the mantissa are signed. Special values are assigned to express the following:

Value	Exponent	Mantissa	Meaning
0x007FFFFE	0	+ (2^23 – 2)	+ INFINITY
0x007FFFFF	0	+ (2^23 – 1)	NaN (Not a Number)
0x00800000	0	– (2^23)	NRes (Not at this Resolution)
0x00800001	0	– (2^23 – 1)	Reserved for future use
0x00800002	0	– (2^23 – 2)	– INFINITY

Table 1.2: Special FLOAT values

2 Service

Requirements in this section are defined as "Mandatory" (M), "Optional" (O), "Excluded" (X), “Not Applicable” (N/A), or "Conditional" (C.n). Conditional statements (C.n) are listed directly below the table in which they appear.

2.1 Service dependencies

GHSS does not depend on other services.

2.2 Bluetooth Core Specification release compatibility

This specification is compatible with Bluetooth Core Specification v4.2 or later [4] that includes the Generic Attribute Profile (GATT).

2.3 Transport dependencies

There are no transport-related dependencies.

2.4 Attribute Protocol error codes

This service references Common Profile and Service Error Codes that are defined in Part B, Section 1 of the Core Specification Supplement (CSS) [3] as listed in the table below. This service also defines Attribute Protocol Application error codes. Table 2.1 lists the set of error codes used in this service.

Name	Error Code	Description
Client Characteristic Configuration Descriptor Improperly Configured	0xFD	Common profile error code defined in CSS
Procedure Already In Progress	0xFE	Common profile error code defined in CSS
Out of Range	0xFF	Common profile error code defined in CSS
Command Not Supported	0x81	Application error code

Table 2.1: Attribute Protocol error codes used by this service

2.5 GATT sub-procedure requirements

Requirements in this section represent a minimum set of requirements for a server. Other GATT sub-procedures may be used if supported by both the client and the server.

Table 2.2 summarizes additional GATT sub-procedure requirements beyond those required by all GATT Servers, as defined in Table 4.1 in Volume 3, Part G of the Bluetooth Core Specification [4].

GATT sub-procedure	Requirements
Exchange MTU	C.1
Read Long Characteristic Values	C.2
Characteristic Value Notification	C.3

GATT sub-procedure	Requirements
Characteristic Value Indication	C.4
Read Characteristic Descriptors	M
Write Characteristic Descriptors	M
Write Characteristic Value	C.5

Table 2.2: GATT sub-procedure requirements

C.1: Mandatory for LE Transport.

C.2: Mandatory if the Health Feature characteristic has a size beyond (ATT_MTU – 3).

C.3: Mandatory if one of the Observation characteristics supports notifications.

C.4: Mandatory if one of the Observation characteristics supports indications or when the Record Access Control Point (RACP) is supported.

C.5: Mandatory when the RACP or the GHS Control Point is supported.

2.6 Declaration

The service UUID shall be set to «Generic Health Sensor Service» as defined in the Bluetooth Assigned Numbers [2].

GHSS shall be instantiated as a «Primary Service».

Only one instance of GHSS is recommended in a device.

2.7 Behavior

GHSS is designed to be implemented on a Personal Health Device (PHD) that generates observations of some health-related phenomenon. A GHSS client collecting these observations is called a collector. This specification distinguishes three categories of observations:

- **Live observations**: observations that are created and communicated by the PHD while connected to a collector.
- **Temporarily stored observations**: observations that are created by a PHD while not connected to a collector or is otherwise not able to send observations to a collector. The observations are stored until they can be sent to a collector. After sending, the PHD deletes the observations.
- **Stored observations**: observations that are created in the past and stored on the PHD for later retrieval by one or more collectors.

A server can choose one of the following scenarios to transfer such observations to clients:

- Support only live observations
- Support only temporarily stored and live observations
- Support only stored data via the RACP

- Support live, temporarily stored, and stored observations – temporarily stored observations go to the first collector that connects and are also stored for later retrieval via RACP by any collector
    - Observations that report an intermediate result of a measurement process may or may not be stored for RACP retrieval.

A server shall transfer observations in the order they are generated, with the oldest being sent first. This requirement applies to both stored observations that are transferred in an RACP procedure as well as to temporarily stored and live observations.

A server might not have the resources to transfer stored observations concurrently with live or temporarily stored observations. In that case, a client should use the mechanisms sequentially (see also Section 3.4).

A GHS server supporting live observations shall support indications or notifications for live observations. A GHS server supporting live observations may support both indications and notifications for live observations.

Because notifications are not reliable, a GHS server should support indications to send episodic live observations for which a reliable transfer is required. Notifications should be supported for observations that are of a temporal nature for which performance is desired, such as sample arrays sent by an electrocardiogram (ECG) device.

Similarly, for stored observations, a GHS server should support indications and notifications, which allows a client to choose indications at the expense of performance to achieve reliable delivery.

When both indications and notifications are enabled for transferring observations, the server should use indications for observations requiring reliable transfer, and the server should use notifications for observations that benefit from higher throughput. The server shall use either indications or notifications but shall not use both to transfer a complete observation. This applies to stored and to live observations.

3 Service characteristics

Requirements in this section are defined as "Mandatory" (M), "Optional" (O), "Excluded" (X), “Not Applicable” (N/A), or "Conditional" (C.n). Conditional statements (C.n) are listed directly below the table in which they appear.

This section defines the characteristic and descriptor requirements. Where a characteristic can be indicated or notified, a Client Characteristic Configuration descriptor must be included in that characteristic as required by [4].

Characteristic / Descriptor Name	Requirement	Mandatory Properties	Optional Properties	Security Permissions
Health Sensor Features	O	Read	Indicate (C.1)	None
Observation Schedule descriptor	O	Read	Write	None
Valid Range and Accuracy descriptor	O	Read	-	None
Live Health Observations	C.2	Indicate or Notify (C.3, Note)	-	None
Stored Health Observations	C.2	Indicate or Notify (C.3, Note)	-	None
RACP	C.4	Write, Indicate	-	None
GHS Control Point	C.5	Write, Indicate	-	None
Observation Schedule Changed	C.6	Indicate	-	None

Table 3.1: Characteristics for use with this service

C.1: Mandatory to support indications of the Features characteristic if the supported features can change (over the lifetime of the device), otherwise Excluded.

C.2: Mandatory to support at least one of the Life Health Observations characteristic or the Stored Health Observations characteristic.

C.3: Mandatory to support indications or notifications. Supporting both is also permitted.

C.4: Mandatory to support the RACP characteristic if the Stored Health Observations characteristic is supported, otherwise Excluded.

C.5: Mandatory to support the GHS Control Point if the Live Health Observations characteristic is supported, otherwise Excluded.

C.6: Mandatory to support the Observation Schedule Changed characteristic if an Observation Schedule descriptor can change, otherwise Excluded.

Note: When a characteristic supports the Indicate and the Notify property, the server may not support enabling indications and notifications simultaneously; see Volume 3, Part G, Section 4.12.3 of the Bluetooth Core Specification [4].

3.1 Health Sensor Features

The Health Sensor Feature characteristic shall be used to describe the supported features of the Generic Health Sensor.

3.1.1 Health Sensor Features format

The Health Sensor Features characteristic has the format shown in Table 3.2.

Field	Requirement	Data Type	Size (in octets)	Description
Flags	M	struct	1	Flags to report the presence of optional fields
Supported Observation Types	M	struct	variable	Mandatory field that reports the number of supported observation types and their codes
Supported Device Specializations	O	struct	variable	Optional field that reports the number of supported device specializations and their codes

Table 3.2: Health Sensor Features characteristic

3.1.1.1 Flags

The bits of the Flags field are defined in Table 3.3.

Bit Number	Definition
0	Supported Device Specializations field present 0 = not present; 1 = present
1-7	Reserved for Future Use

Table 3.3: Flags field

3.1.1.2 Supported Observation Types

The Supported Observation Types field reports the observation types that a Generic Health Sensor can generate. These observation types are encoded using a code from the IEEE 11073-10101 nomenclature specification [5]. The structure of this field is shown in Table 3.4.

Field	Requirement	Data Type	Size (in octets)	Description
Count	M	uint8	1	The number of supported observation types

Field	Requirement	Data Type	Size (in octets)	Description
Codes	M	uint32[Count]	Count * 4	The concatenation of the 4-byte codes for the supported observations

Table 3.4: Supported Observation Types field

An example of the Supported Observation Types field for a pulse oximeter sensor that reports blood oxygen saturation and pulse rate observations is shown in Table 3.5. This example uses the following 4-byte Medical Device Communication (MDC) codes from IEEE 11073-10101 [5]:

MDC_PULS_OXIM_SAT_O2 = 0x00024BB8

MDC_PULS_OXIM_PULS_RATE = 0x0002481A

Field	Data Type	Size (in octets)	Value (hex)
Count	uint8	1	02
Codes	struct	8	B8 4B 02 00 1A 48 02 00

Table 3.5: Example of Supported Observation Types field

Note: The byte-order in the table is little-endian and the byte order in [5] is big-endian.

3.1.1.3 Supported Device Specializations

The Supported Device Specializations field reports the supported ACOM device specializations that a Generic Health Sensor can generate. These device specializations are encoded using a code from the IEEE 11073-10101 nomenclature specification [5] and a version number. The structure of this field is shown in Table 3.6.

An ACOM device specialization defines a coherent set of observation types that a sensor implementing that specialization supports. The device specialization defines which of these observation types the PHD shall support and which observation types the PHD may optionally support.

Because MDC codes for device specializations are coming from the infrastructure partition, partition 8 in IEEE 11073-10101 [5], these MDC codes can be encoded using a 2-byte term code.

Field	Requirement	Data Type	Size (in octets)	Description
Count	M	uint8	1	The number of supported observation types
Device Specializations	M	struct	Count * 3	The concatenation of tuples consisting of the 2-byte codes for the supported device specializations from the infrastructure partition, partition 8, in IEEE 11073-10101 [5] and their 1-byte version numbers

Table 3.6: Supported Device Specializations field

An example of the Supported Device Specializations field for a blood pressure monitor is shown in Table 3.7. This example uses the following MDC code from [5] without the first two bytes that indicate partition 8:

Code = MDC_DEV_SPEC_PROFILE_BP = 0x00081007; Version = 0x01

Field	Data Type	Size (in octets)	Value (hex)
Count	uint8	1	01
Device Specializations	struct	3	07 10 01

Table 3.7: Example of Supported Device Specializations field

3.1.2 Health Sensor Features behavior

When read or indicated, the Health Sensor Features characteristic gives a value that is used by a client to determine the supported features of the Generic Health Sensor. Indications shall be supported if the supported features of the Generic Health Sensor can change.

When the Client Characteristic Configuration Descriptor is configured for indications and the supported features of a Generic Health Sensor have changed, then the Health Sensor Features characteristic shall be indicated to connected clients and to any bonded clients after reconnection.

Note: The indication of the Health Sensor Features characteristic can only carry the first (ATT_MTU – 3) bytes of the characteristic value. A long read is needed by a client to read subsequent bytes.

3.2 Live Health Observations

The Live Health Observations characteristic is used by the Generic Health Sensor to report generated observations to a client using indications or notifications.

These observations are reported live when they are generated from measurements, or when they are first temporarily stored and reported when a connection with a client is established. Temporarily stored observations shall include a time stamp.

The Live Health Observations characteristic supports the different ACOM observation class types to cover the different value types ACOM observations can have.

A 1-byte segmentation scheme is used to support values that do not fit in a single Maximum Transfer Unit (MTU).

3.2.1 Live Health Observations format

The Live Health Observation characteristic has the format defined in Table 3.8.

Field	Requirement	Data Type	Size (in octets)	Description
Segmentation Header	M	struct	1	See Table 3.9.
Health Observation Body Segment	M	struct	variable	A segment of a Health Observation Body structure as defined in Table 3.10. Segmentation is defined in Section 3.2.2.1.

Table 3.8: Live Health Observations characteristic

3.2.1.1 Segmentation Header

Table 3.9 defines the values of the Segmentation Header field. The server shall use these values to provide information to the client on the segments to concatenate to get a complete Health Observation Body structure.

Bit number	Definition
0	**First Segment:** The characteristic contains the first segment of content that should be concatenated by the client. 0 = False 1 = True
1	**Last Segment:** The characteristic contains the last segment of content that should be concatenated by the client. 0 = False 1 = True
2-7	Rolling Segment Counter: 0 to 63 The Rolling Segment Counter is incremented per segment during a connection. If the Rolling Segment Counter is equal to a value of 63, then the counter rolls over to 0 when it is next incremented.

Table 3.9: Segmentation Header structure

3.2.1.2 Health Observation Body

The structure of the Health Observation Body is shown in Table 3.10.

Field	Requirement	Data Type	Size (in octets)	Description
Observation Class Type	M	uint8	1	The ACOM Observation Class Type as defined in Table 3.11.
Length	M	uint16	2	The length in bytes of the Health Observation Body structure.
Flags	M	struct	2	Flags indicating the presence of optional attributes as defined in Table 3.12.
Observation Type	O	uint32	4	An MDC code (e.g., for SpO2) representing the type of observation. Mandatory for all single observations but not for observation bundles. When observations are bundled, this field may be factored out if it is common to all observations in the bundle (see Sections 3.2.1.5 and 3.2.1.15.9). Presence indicated by flag.
Time Stamp	O	Elapsed Time	9	The time the observation was observed (see Section 3.2.1.6). Presence indicated by flag.
Measurement Duration	O	FLOAT	4	The duration of the measurement in seconds (see Section 3.2.1.7). Presence indicated by flag.
Measurement Status	O	struct	2	Measurement status bits as defined in Section 3.2.1.8. Presence indicated by flag.
Observation Id	O	uint32	4	Instance identifier for unique referencing to an Observation (see Section 3.2.1.9). Presence indicated by flag.
Patient	O	uint8	1	Patient or user identifier (see Section 3.2.1.10). Presence indicated by flag.
Supplemental Information	O	struct	variable	Information that helps in understanding the observation (see Section 3.2.1.11). Presence indicated by flag.
Derived From	O	struct	variable	References to other observations (see Section 3.2.1.12). Presence indicated by flag.

Field	Requirement	Data Type	Size (in octets)	Description
Is Member Of	O	struct	variable	References to other observations (see Section 3.2.1.13). Presence indicated by flag.
TLVs	O	struct	variable	Type-Length-Value (TLV) extension attributes (see Section 3.2.1.14). Presence indicated by flag.
Observation Value	M	struct	variable	Mandatory field. Structure depends on the Observation Class Type (see Section 3.2.1.15).

Table 3.10: Health Observation Body structure

3.2.1.3 Observation Class Type

Table 3.11 lists the supported ACOM Observation Class Type values.

Observation Class Type	Value	Description
Numeric observation	1	The numeric observation value is defined in Section 3.2.1.15.1.
Simple discrete observation	2	The simple discrete observation value is defined in Section 3.2.1.15.2.
String observation	3	The string observation value is defined in Section 3.2.1.15.3.
Sample array observation	4	The sample array observation value is defined in Section 3.2.1.15.4.
Compound discrete event observation	5	The compound discrete observation value is defined in Section 3.2.1.15.5.
Compound state/event observation	6	The compound state/event observation value is defined in Section 3.2.1.15.6.
Compound observation	7	The compound observation value is defined in Section 3.2.1.15.7.
TLV-encoded observation	8	The TLV-encoded observation value is defined in Section 3.2.1.15.8.
Observation bundle	255	The observation bundle value is defined in Section 3.2.1.15.9.
Other values are RFU	0, 9-254	RFU values shall not be used by implementations claiming conformance to this version of GHSS.

Table 3.11: Observation Class Types

The Observation Class Type determines the type of value reported by the observation. These value types are defined in Section 3.2.1.15.

3.2.1.4 Flags

The Flags field bits and their functions are defined in Table 3.12.

The Flags field values reflect the presence of the optional fields. When the Flags field is present, the corresponding bit of the Flags field shall be set to a value of 1, otherwise the corresponding bit of the Flags field shall be set to a value of 0.

Bit	Bit Name
0	Observation Type present
1	Time Stamp present
2	Measurement Duration present
3	Measurement Status present
4	Observation Id present
5	Patient present
6	Supplemental Information present
7	Derived From present
8	Is Member Of present
9	TLVs present
10-15	RFU

Table 3.12: Flags field

3.2.1.5 Observation Type

The Observation Type field contains a 4-byte MDC code from IEEE 11073-10101 [5] that uniquely defines the type of the observation. A few examples are:

- MDC_PULS_OXIM_SAT_O2 = 0x00024BB8

  This code identifies a blood oxygen saturation observation with pulse oximetry.

- MDC_PULS_OXIM_PULS_RATE = 0x0002481A

  This code identifies a pulse rate observation with pulse oximetry.

The Observation Type field is mandatory for single observations that are not part of a bundle, and the field is optional for observation bundles. For single observations in bundles, either the bundle contains the Observation Type field that applies to all contained single observations, or each single observation contains the Observation Type field.

The code for the Observation Type can come from the metrics partition (partition 2), the PHD disease management partition (partition 128), the PHD health fitness partition (partition 129) or the PHD aging independently partition (partition 130) of IEEE 11073-10101 [5], but codes from other partitions are not excluded. Private codes can also be used and can come from the private partition or from private code blocks in other partitions.

3.2.1.6 Time Stamp

The Time Stamp field contains an Elapsed Time structure as defined in GSS [1], representing the time the observation was observed.

When there is a Measurement Duration, the Time Stamp field represents the start of the measurement.

3.2.1.7 Measurement Duration

The optional Measurement Duration field contains the duration of the measurement in seconds, reported as a floating number.

3.2.1.8 Measurement Status

The Measurement Status field is defined as a series of Boolean conditions. See Table 3.13 for the conditions being reported.

Bit	Condition	Description
0	Invalid	The observation is invalid.
1	Questionable	The observation is questionable.
2	Not available	No actual observation can be generated.
3	Calibrating	The observation is generated during a calibration procedure.
4	Test data	The observation contains test data.
5	Early estimate	The observation contains an early estimate (of a final value).
6	Manually entered	The observation value is manually entered.
7	Setting	The observation value is a device setting.
14	Threshold error	The (numeric) observation value is outside the boundaries. The boundaries exceeded may be reported as a TLV attribute identified by the MDC Code MDC_ATTR_LIMIT_CURR.
15	Thresholding disabled	The observation value thresholds are not checked. If bit 15 is set, bit 14 shall not be set.
8-13	RFU	Reserved for Future Use

Table 3.13: Measurement Status bits

3.2.1.9 Observation Id

The Observation Id field contains a value that can be used in other observations to reference this specific observation.

The value shall be unique in the context of its usage, which is defined as the set of observations transferred from the server to a client during a connection. When observations have references to other observations, these referenced observations shall be communicated first.

The server can use a simple scheme to assign Observation Id values by starting from a value of 0 and incrementing by one for each next Observation Id being used in a generated observation during a connection. Other schemes that assign unique Observation Ids during a connection are allowed as well.

The Observation Id field shall be absent at the top level of an observation bundle (i.e., when the Observation Type is set to Observation bundle, the Observation Id present flag is set to false). The observation values contained in the bundle can have a unique Observation Id (see Section 3.2.1.15.9).

3.2.1.10 Patient

This optional Patient field contains a local identification of the patient or user using the PHD.

3.2.1.11 Supplemental Information

This optional Supplemental Information field contains supplemental information that can help in understanding the observation. The field's structure is defined in Table 3.14. The field can be used to report the body location of the sensor, the meal context of an observation, or other relevant aspects. Only information that can be expressed by an MDC code can be reported.

Field	Requirement	Data Type	Size (in octets)	Description
Count	M	uint8	1	The number of MDC codes present in the Codes field
Codes	M	uint32[Count]	Count * 4	A series of MDC codes

Table 3.14: Supplemental Information structure

An example value for a pulse-oximeter reporting a spot-check is shown in Table 3.15.

Field	Data Type	Size (in octets)	Value
Count	uint8	1	01
Codes	variable	4	3C 4C 02 00 (MDC_MODALITY_SPOT)

Table 3.15: Supplemental Information example

3.2.1.12 Derived From

The Derived From field contains references to other observations from which this observation is derived. To reference an observation, its Observation Id value is used. The structure of this field is shown in Table 3.16.

Field	Requirement	Data Type	Size (in octets)	Description
Count	M	uint8	1	The number of Observation Ids present in the Observation Ids field
Observation Ids	M	uint32[Count]	Count * 4	A series of Observation Id field values

Table 3.16: Derived From and Is Member Of structure

3.2.1.13 Is Member Of

The Is Member Of field is present in observations that are part of a group of observations. An example would be an exercise session on a fitness machine that reports the type of the session and that includes the individual heart rate observations taken during the session. The Is Member Of field contains the reference to the group observation. The structure of this field is shown in Table 3.16. In most cases, an observation is a member of one group, but the structure allows for membership in multiple groups.

3.2.1.14 TLVs

ACOM allows new attributes to be added to defined classes and also allows new classes to be added to the model. To support a new type of attribute in an observation, a Type-Length-Values field (TLV) is supported.

The structure of this field is shown in Table 3.18.

Field	Requirement	Data Type	Size (in octets)	Description
Number Of TLV Attributes	M	uint8	1	N, the number of TLV attributes in this field.
Value	M	variable	variable	The concatenation of the N TLV attributes of the field. The structure of a TLV attribute is defined in Table 3.18.

Table 3.17 TLVs structure

For the Type of a single TLV, an MDC code is used that can come from partition 1 of IEEE 11073-10101 [5], the Object-oriented partition, and that has a Reference Id starting with “MDC_ATTR_”. Codes from other partitions, including private codes for manufacturer-specific extensions, can also be used.

Each TLV field contains a Format Type with a value as defined in Section 2.6.1 GATT Format Types of the Assigned Numbers [2]. The Format Type determines the encoding of the Encoded Value field. There are Format Types for the encoding of numeric values and strings of various sizes. When the Format Type has a value of 0x1B, the encoded value is an opaque structure that is not further defined by GHSS. When

the Format Type has a value of 0x1C, the encoded value follows the MDER encoding rules for an ASN.1 defined value as defined in Annex F of [9].

The structure of the TLV field is shown in Table 3.18.

Field	Requirement	Data Type	Size (in octets)	Description
Type	M	uint32	4	The MDC code from IEEE 11073-10101 [5] identifying the type of the TLV attribute.
Length	M	uint16	2	The length of the Encoded Value field in octets. The Format Type is not counted in the length.
Format Type	M	uint8	1	A Format Type as defined in Section 2.4.1 GATT Format Types of [2].
Encoded Value	M	variable	Length	The value of the observation attribute identified by the Type encoded according to the Format Type.

Table 3.18: TLV structure

3.2.1.15 Observation value

The Observation value depends on the Observation Class Type. The subsections of this section cover the structure for each Observation Class Type.

3.2.1.15.1 Numeric observation value

A numeric observation is used to report a measured quantity using a numeric value and a unit of the quantity. An example would be a temperature measurement of 37.5 degrees Celsius.

The Numeric observation value is a structure defined in Table 3.19.

Field	Requirement	Data Type	Size (in octets)	Description
Unit Code	M	uint16	2	A 2-byte MDC code from partition 4 of IEEE 11073-10101 [5] that identifies the unit of Value
Value	M	FLOAT	4	The numeric value of the observation encoded as an IEEE 11073 32-bit FLOAT

Table 3.19: Numeric observation value structure

3.2.1.15.2 Simple discrete observation value

A simple discrete observation is used to report a value from a discrete set of possibilities. An example would be the observed body position of a person with sitting, standing, or laying as possible values.

The Simple discrete observation value is a structure defined in Table 3.20.

Field	Requirement	Data Type	Size (in octets)	Description
Value	M	uint32	4	The MDC code that identifies the value of this observation

Table 3.20: Simple discrete observation value structure

3.2.1.15.3 String observation value

A string observation is used to report values that are best represented as text. Strings may lead to interoperability issues because of language dependencies and should be avoided when other class types can be used. An example would be to report the user's gender as a string with a value of "Man" (Dutch for "male").

The String observation value is a structure defined in Table 3.21.

Field	Requirement	Data Type	Size (in octets)	Description
Length	M	uint16	2	Length of the string
Value	M	utf8s	variable	UTF-8 string of the given Length

Table 3.21: String observation value structure

3.2.1.15.4 Sample array observation value

A Sample array observation is used to report a series of samples from a measured quantity. An example would be an electrocardiogram (ECG) real-time wave form.

The Sample array observation value is a structure defined in Table 3.22.

Field	Requirement	Data Type	Size (in octets)	Description
Unit Code	M	uint16	2	The 2-byte MDC code from partition 4 of IEEE 11073-10101 [5].
Scale Factor	M	FLOAT	4	The scaling factor to be applied to the samples.
Offset	M	FLOAT	4	The offset to be applied to the samples.
Sample Period	M	FLOAT	4	The duration in seconds.
Number Of Samples Per Period	M	uint8	1	The number of samples per period. This can be used to support multi-dimensional sample arrays.
Bytes Per Sample	M	uint8	1	The number of bytes for one sample. The recommended values are 1, 2, or 4.

Field	Requirement	Data Type	Size (in octets)	Description
Number Of Samples	M	uint32	4	The total number of samples.
Scaled Samples	M	variable	Number of samples * Bytes per sample	The scaled samples.

Table 3.22: Sample array observation value structure

The actual sample values can be calculated as follows:

For each i in the range of 0...Number Of Samples:

Sample[i] = Scaled Sample[i] * Scale Factor + Offset

3.2.1.15.5 Compound Discrete Event observation value

A Compound Discrete Event observation value is used to report a set of MDC codes representing discrete events. An example would be reporting a blood pressure measurement status of “cuff too loose” combined with “body movement detected”.

The Compound Discrete Event observation value is a structure defined in Table 3.23.

Field	Requirement	Data Type	Size (in octets)	Description
Number Of Components	M	uint8	1	The number of discrete events in this observation
Value	M	variable	4* Number Of Components	The concatenation of the MDC codes for the event values

Table 3.23: Compound Discrete Event observation value structure

3.2.1.15.6 Compound State/Event observation (bit string) value

A Compound State/Event observation reports a set of Boolean states or events encoded as bits as defined by the type of the observation in IEEE 11073-10101.

The Compound State/Event observation value is a structure defined in Table 3.24.

Field	Requirement	Data Type	Size (in octets)	Description
Size	M	uint8	1	N is the number of bytes used for the state/event bits. The value of this field determines the size of the other fields of this structure.
Supported Or Unsupported Mask Bits	M	struct	N	N*8 bits indicates if the (binary) state or event is supported in the observation. 1 = supported, 0 = unsupported
State Or Event Mask Bits	M	struct	N	N*8 bits indicates if the bit represents a state or an event. 1 = state, 0 = event
Value	M	struct	N	N*8 bits indicates if the state is valid or if the event occurred in this observation. When supported: 1 = state is valid or event occurred 0 = state is not valid or event did not occur

Table 3.24: Compound State/Event observation value structure

3.2.1.15.7 Compound observation value

A compound observation is used to report a set of closely related observation components that need to be interpreted together. An example would be reporting a discrete posture of a patient and the amount of time the patient was observed in that posture.

The Compound observation value is a structure defined in Table 3.25.

Field	Requirement	Data Type	Size (in octets)	Description
Number Of Components	M	uint8	1	N, the number of components in this observation
Value	M	Observation-component*N	variable	The concatenation of the components of the observation

Table 3.25: Compound observation value structure

3.2.1.15.7.1 Observation component

The Observation component is a structure defined in Table 3.26.

Field	Requirement	Data Type	Size (in octets)	Description
Component Type	M	uint32	4	Term code, a 4-byte MDC code from IEEE 11073-10101 [5] that uniquely defines the type of the component.
Component Value Type	M	uint8	1	1-byte enumeration value that indicates the value type of the component (see Table 3.27).
Value	M	struct	variable	The value of the component. The structure of the value depends on the type of the component as defined in Table 3.27.

Table 3.26: Observation component structure

Component Value Type	Definition	Description
1	Numeric	The Numeric observation value is a structure defined in Table 3.19.
2	Simple discrete	The Simple discrete observation value is a structure defined in Table 3.20.
3	String	The String observation value is a structure defined in Table 3.21.
4	Sample array	The Sample array observation value is a structure defined in Table 3.22.
5	Compound discrete	The Compound discrete observation value is a structure defined in Table 3.23.
6	Compound state/event	The Compound state/event observation value is a structure defined in Table 3.24.
0, 7-255	RFU	Not defined.

Table 3.27: Observation Component Value Types

3.2.1.15.8 TLV-encoded observation value

A TLV-encoded observation can be used to encode an observation with attributes that are all identified by IEEE 11073-10101 MDC codes. A TLV-encoded observation value is a set of TLV tuples.

The TLV-encoded observation value follows the structure defined in Table 3.17.

3.2.1.15.9 Observation bundle value

An Observation bundle value can be used to report a set of observations that may have references to each other or that share common attributes. The Observation bundle value should also be used for live observations that are generated at the same time but that carry no time stamp. Bundling observations allows the receiving client to assign the same time stamp of the moment of reception to the bundled observations. A server may use the Observation bundle to reduce the size of the data needed to communicate many observations with common attributes.

The Observation bundle value is a structure defined in Table 3.28.

Field	Requirement	Data Type	Size (in octets)	Description
Number Of Observations	M	uint8	1	N, the number of observations in this bundle.
Value	M	variable	variable	The concatenation of the N bundled Health Observation bodies. Each Observation shall be encoded as defined in Section 3.2.1.2, leaving out the attributes that have been factored out. An observation bundle shall not be nested within another bundle.

Table 3.28: Observation bundle value structure

3.2.2 Live Health Observations behavior

The Live Health Observations characteristic is used by the Generic Health Sensor to send generated ACOM observations to a client.

3.2.2.1 Segmentation

If the total size of a generated observation is greater than (ATT_MTU-4), then multiple segments shall be sent to transfer a single Health Observation Body (the data to be transported).

When the Live Health Observations characteristic is configured for indications or notifications via the Client Characteristic Configuration descriptor, when the server has received a command to start sending live observations (see Section 3.5), and when a generated health observation is available, then this characteristic shall be indicated or notified per segment as described below.

The number of segments that needs to be sent to convey the data to be transported shall be calculated by dividing the size of the data to be transported by (ATT_MTU-4) rounded up to the nearest integer. N represents this number in the steps below:

1. If N=1, then the First Segment and Last Segment bits of the Segmentation Header field shall both be set to a value of 1.
2. If N>1, then the First Segment bit of the Segmentation Header field shall be set to a value of 1, and the Last Segment bit shall be set to a value of 0.
3. The Rolling Segment Counter shall be set to the current value maintained by the server.
4. Up to (ATT_MTU-4) octets of the data to be transported shall be used to fill the characteristic message.
5. The Live Health Observations characteristic shall be indicated or notified.
6. If the value of the Last Segment bit is 1, then end the procedure at this point, otherwise continue to step 7.
7. The First Segment bit of the Segmentation Header field shall be set to a value of 0.
8. If the number of octets remaining to be sent can be sent in one more notification of the Live Health Observations characteristic, then the Last Segment bit of the Segmentation Header field shall be set to a value of 1, otherwise it shall remain set to a value of 0.
9. The value of the Rolling Segment Counter shall be incremented by 1.

10. The next octets of the contents of the data to be sent shall be used to fill the characteristic message, in order, consisting of up to (ATT_MTU -4) octets.
11. Repeat from step 4.

When the service is initialized (e.g., upon connection to the client), the Rolling Segment Counter may be set to any number in the range of 0 to 63. The Rolling Segment Counter should then be incremented continuously without being reset each time the above procedure is used. When the Rolling Segment Counter reaches a value of 63, the Rolling Segment Counter shall roll over to 0 the next time it is incremented.

3.2.2.2 Observation reporting

A GHS Server may follow one of the following strategies to report observations:

- Only report live and optionally temporarily stored observations
  - With this strategy, there is no need to use an RACP and a Stored Health Observations characteristic, and observations are deleted from the server after retrieval. Observations are sent to only one client using indications or notifications of the Live Health Observations characteristic.
- Only report stored observations
  - With this strategy, an RACP is used to support retrieval of all observations under client control. Observations generated during a connection are not immediately reported. Multiple clients can retrieve all observations.
- Store generated observations and report live observations when connected
  - With this strategy, an RACP is used to support retrieval of observations under client control. Multiple clients can retrieve all observations. In addition, observations generated during a connection are immediately reported.
- Other strategies may be possible as well, where some types of observations are stored, and other observations are only sent live. This could be a good strategy for reporting the intermediate cuff pressure during a blood pressure measurement that does not have much value as a stored observation for later retrieval. The precise strategy of reporting observations is chosen by the implementer.

3.3 Stored Health Observations

The Stored Health Observations characteristic is used to report Stored Health Observations to clients using RACP procedures.

3.3.1 Stored Health Observations format

The Stored Health Observation characteristic has the format as defined in Table 3.29.

Field	Requirement	Data Type	Size (in octets)	Description
Segmentation Header	M	uint8	1	See Table 3.9

Field	Requirement	Data Type	Size (in octets)	Description
Stored Health Observations Segment	M	struct	variable	A segment of a Stored Health Observation Body structure as defined in Table 3.30

Table 3.29: Stored Health Observation characteristic

3.3.1.1 Stored Health Observations Body

The structure of the Stored Health Observations Body field is shown in Table 3.30.

Field	Requirement	Data Type	Size (in octets)	Description
Record Number	M	uint32	4	The record number of the stored observation or stored observation bundle
Health Observation Body	M	struct	variable	A Health Observation Body as defined in Table 3.10

Table 3.30: Stored Health Observations Body

This structure adds a 4-byte record number to the Health Observation Body structure. The record number is used for retrieval of the stored observations using the RACP procedures.

The Record Number should start at a value of 0 for the first generated stored observation and should be increased by one for each subsequently generated observation. The record number should be reset to a value of 0 after 2^32 records.

3.3.2 Stored Health Observations behavior

The Stored Health Observations characteristic shall be indicated or notified to report stored observations, in a segmented fashion, as the response to an RACP command from a client to retrieve a selection of stored records. Stored observations shall have a time stamp.

The segmentation scheme used to report the Stored Health Observations characteristic is the same scheme that is used for the Live Health Observations characteristic (see Section 3.2.2.1).

3.4 Record Access Control Point (RACP)

A Record Access Control Point (RACP) characteristic is included as an optional characteristic with associated procedures.

The RACP allows one or more clients to retrieve Stored Health Observations from the server.

A server may support both retrieval of Stored Health Observations and sending Live and Temporarily Stored Health Observations, but might not be able to support both at the same time (see Section 3.4.4 for the server behavior in that situation).

If the User Data Service (UDS) [8] is used in combination with GHSS, then the Stored Health Observations will be retrievable on a per-user basis (see GHSP [7] for more details).

Stored Health Observations can be retrieved using record numbers. Time-based filtering may be supported as well, but it is not considered the most robust approach because time faults may cause discontinuities in the timeline of the server.

See GSS [1] for the definition of the RACP characteristic.

3.4.1 Record definition

In the context of the Generic Health Service, a record is a Stored Health Observation.

3.4.2 RACP Op Codes, operators, and operands requirements

Table 3.31 and Table 3.32 show the detailed requirements on RACP Op Codes, Operators, and Operands for Operations and Responses. Op Codes that are not present in Table 3.31 or Table 3.32 shall not be supported. The Op Codes for Combined Report and Report Number of Stored Records have the same requirements that are combined in one row.

Commands						
Op Code	**Op Code Requirement**	**Operator**	**Operator Requirement**	**Operand**		**Operand Requirement**
				Filter Type	**Filter Parameters**	
Combined Report, Report Number of Stored Records	M	All records	M	No Operand Used		N/A
		Less than or equal to	O	Record Number	<maximum filter value>	C.1
				Time Filter	<maximum filter value>	O
		Greater than or equal to	M	Record Number	<minimum filter value>	M
				Time Filter	<minimum filter value>	O
		Within range of (inclusive)	O	Record Number	<minimum filter value>, <maximum filter value>	C.1
				Time Filter	<minimum filter value>, <maximum filter value>	O
		First record	O	No Operand Used		N/A
		Last record	O	No Operand Used		N/A
Delete Stored Records	O	All records	C.2	No Operand Used		N/A
		Less than or equal to	O	Record Number	<maximum filter value>	C.1
				Time Filter	<maximum filter value>	O
		Greater than or equal to	O	Record Number	<minimum filter value>	C.1
				Time Filter	<minimum filter value>	O

Commands						
Op Code	**Op Code Requirement**	**Operator**	**Operator Requirement**	**Operand**		**Operand Requirement**
				Filter Type	**Filter Parameters**	
		Within range of (inclusive)	O	Record Number	<minimum filter value>, <maximum filter value>	C.1
				Time Filter	<minimum filter value>, <maximum filter value>	O
		First record	O	No Operand Used		N/A
		Last record	O	No Operand Used		N/A
Abort Operation	O	Null (0x00)	C.2	No Operand Used		N/A

Table 3.31: RACP Procedure requirements for Operations

C.1: Mandatory to support the Record Number Operand for this Operator if this Operator is supported (see also Section 3.4.2.1).

C.2: Mandatory to support this Operator for this Op Code if this Op Code is supported.

Responses					
Op Code	**Op Code Requirement**	**Operator**	**Operator Requirement**	**Operand**	**Operand Requirement**
Combined Report Response, Number of Stored Records Response	M	Null (0x00)	M	uint32 containing number of records	M
Response Code	M	Null (0x00)	M	Request Op Code, Response Code Value	M

Table 3.32: RACP Procedure requirements for Responses

3.4.2.1 Further RACP procedure details

The same Operator and Operand combinations shall be supported for the Combined Report and the Report Number of Stored Records Op Code. The supported combinations for the Delete Stored Records Op Code may be different.

A server that is capable of storing a large number of stored observations should support the “Within range of” RACP Operator (see Table 3.31) to avoid having to transfer a large number of records.

Filter parameters for Filter Type Record Number are uint32 record number values.

Filter parameters for Filter Type Time are 6-byte time values as specified in the Elapsed Time in GSS [1].

When a Filter Type is supported, the same combinations of Operand and Operator shall be supported as for the Record Number Filter Type.

Response Code values for the RACP can be found in GSS [1].

3.4.2.2 Filter types

The RACP shall support filtering on record numbers.

The RACP may support filtering on time as well. The server will return records that have a time stamp that matches the filter conditions. The records are sent to the client via notifications on the Stored Health Observation characteristic.

The filter types are uint8 values as listed in Table 3.33.

Operand Filter Type Value	Filter Type Description
0x00	Reserved for Future Use
0x01	Record Number
0x02	Time (optional)
0x03 – 0xFF	Reserved for Future Use

Table 3.33: Operand filter type values

The Time filter is provided to allow a mechanism for reporting records with a given date or time (or a range of dates and times). Because the server time might not be monotonically increasing, the records returned might have non-increasing record numbers.

The time values used for filtering are the uint48 values used for the Time Value in the Elapsed Time characteristic from GSS [1]. Filtering on time takes only the time value into consideration and ignores the value of flags or an offset in the time stamp of an observation.

3.4.3 RACP behavior

The RACP can be used to retrieve or delete Stored Health Observations. Procedures to do this are triggered by a write to the RACP characteristic that includes an Op Code specifying the operation as defined in Table 3.32.

In a multiple-bond case, the handling of the Control Point shall be consistent across all bonds (i.e., there is a single database that is shared by all collectors).

In case the User Data Service is used in combination with GHS, there is a database (set of records) per user ID value for each registered user and one other database (set of records) for all non-registered user ID values.

3.4.3.1 Combined Report procedure

When the Combined Report Op Code is written to the RACP, then the server shall indicate or notify records using the Stored Health Observations characteristic. When all records for a given request have been indicated or notified by the server, then the server shall indicate the RACP with a Combined Report Response Op Code and the Operand set to the number of records reported (see Record Access Control Point in GSS [1]).

The server shall report the records in the order they were created.

If during the record transfer a new record becomes available (i.e., after the Combined Report procedure is initiated), then the server may include this new record in the measurement transfer.

If the server does not locate any records matching the conditions set by the Operator and, when present, that the Operand used, then the server shall indicate the RACP with a Response Code Op Code and a Response Code Value in the Operand set to No Records Found (see RACP in GSS [1]).

If the operation results in an error condition, then the error condition shall be indicated using the Response Code Op Code and the appropriate Response Code Value in the Operand (see Section 3.4.4).

If the server needs to interrupt its data transfer before completion for any reason except in the event of an Abort Operation request, then the server shall indicate the RACP with a Response Code Op Code and a Response Code Value in the Operand set to Procedure not completed (see RACP in GSS [1]). In the event of an Abort Operation command, the procedure terminates immediately without the RACP indicating the Response Code Op Code for this procedure.

3.4.3.2 Report Number of Stored Records procedure

When the Report Number of Stored Records Op Code is written to the RACP, then the server shall calculate and respond with a record count in uint32 format. The response is indicated using the Number of Stored Records Response Op Code.

If the server does not locate any records matching the filter criteria of the request, then the server shall indicate the RACP with a Number of Stored Records Response Op Code and the Operand set to a value of 0x0000.

If the operation results in an error condition, then the error condition shall be indicated using the Response Code Op Code and the appropriate Response Code Value in the Operand (see Section 3.4.4).

3.4.3.3 Delete Stored Records procedure

This section is only applicable if the server supports the Delete Stored Records procedure.

When the Delete Stored Records Op Code is written to the RACP, then the server shall delete all stored observations matching the filter criteria. Deletion of records may be a permanent deletion of records from the database. The server shall indicate this characteristic with a Response Code Value of Success if the records were successfully deleted from the database (see RACP in GSS [1]).

Deleting stored observations might lead to holes in the Record number continuum. The server should not renumber records after deleting some records because this makes it more complex for a client to retrieve newly generated records.

If the server does not locate any records matching the filter criteria of the request, then the server shall indicate the RACP with a Response Code Op Code and Response Code Value in the Operand set to No Records Found.

If the operation results in an error condition, then the error condition shall be indicated using the Response Code Op Code and the appropriate Response Code Value in the Operand (see Section 3.4.4).

3.4.3.4 Abort Operation procedure

This section is only applicable if the server supports the Abort Operation procedure.

When the Abort Operation Op Code is written to the RACP, then the server shall stop any RACP procedures that are in progress and shall make a best effort to stop sending any further data.

When all RACP procedures have been stopped, then the server shall indicate the RACP with a Response Code Op Code and a Response Code Value in the Operand set to Success (see RACP in GSS [1]).

If the operation results in an error condition, then the error condition shall be indicated using the Response Code Op Code and the appropriate Response Code Value in the Operand (see Section 3.4.4).

3.4.4 RACP error handling procedures

If the server is unable to complete a procedure defined in Section 3.4.3 for any reason not covered elsewhere in this section, then the server shall indicate the RACP with a Response Code Op Code and a Response Code Value in the Operand set to Procedure not completed (see RACP in GSS [1]).

If the server is unable to process the Abort Operation procedure for any reason not stated elsewhere in this section, then the server shall indicate the RACP with a Response Code Op Code and Response Code Value in the Operand set to Abort unsuccessful (see RACP in GSS [1]).

If the server is unable to execute an RACP procedure because live observations are being transferred and the server cannot support transfers of live or temporarily stored observations in parallel with executing RACP procedures, then the server shall indicate the RACP with a Response Code Op Code and a Response Code Value in the Operand set to Server Busy.

If a request with an Op Code other than Abort Operation is written to the RACP while the server is performing a previously triggered RACP operation (i.e., resulting from invalid collector behavior), then the server shall return an error response with the Common Profile and Service error code of Procedure Already In Progress, 0xFE (see [4]).

If the Op Code that was written to the RACP requests record notifications and the Client Characteristic Configuration descriptor is not configured for notifications, then the server shall return an error response with the Common Profile and Service error code of Client Characteristic Configuration Descriptor Improperly Configured, 0xFD (see [4]).

If the Operator that was written to the RACP is not supported by the server, then the server shall indicate the RACP with a Response Code Op Code and a Response Code Value in the Operand set to Operator Not Supported (see RACP in GSS [1]).

If the Operator that was written to the RACP is invalid, then the server shall indicate the RACP with a Response Code Op Code and a Response Code Value in the Operand set to Invalid Operator (see RACP in GSS [1]).

If the Op Code that was written to the RACP characteristic is not supported by the server, then the server shall indicate the RACP with a Response Code Op Code and a Response Code Value in the Operand set to Op Code Not Supported (see RACP in GSS [1]).

If an Operand that was written to the RACP characteristic is not supported by the server, then the server shall indicate the RACP with a Response Code Op Code and a Response Code Value in the Operand set to Operand Not Supported (see RACP in GSS [1]).

3.4.5 RACP procedure timeout and failure

In the context of the RACP characteristic, a procedure is started when a write to the RACP characteristic is successfully completed. When a procedure is complete, then the server indicates the RACP characteristic with the Op Code set to the corresponding Response Code.

An RACP procedure may consist of multiple characteristic indications or notifications of the Stored Health Observations Characteristic followed by an indication of the RACP. When the server transmits an indication of the RACP characteristic, then the response is considered to have timed out if the acknowledgement is not received within the procedure timeout, defined as 30 seconds. If a timeout occurs, then the server shall stop sending any further indications and notifications related to the operation and consider the procedure to have failed.

If the connection to the collector is lost, then the procedure shall be considered to have failed and shall not resume upon the next connection.

3.5 GHS Control Point

The GHS Control Point is used to start and stop the sending of live and temporarily stored observations to a client via indications or notifications of the Live Health Observations characteristic during a connection.

3.5.1 GHS Control Point format

The GHS Control Point characteristic has the format shown in Table 3.34.

Field	Requirement	Data Type	Size (in octets)	Description
Op Code	M	uint8	1	The code for the control point command or the response (see Table 3.35)

Table 3.34: GHS Control Point characteristic

GHS Control Point Op Codes	Value	Description
Start sending live observations	0x01	Command to start sending live observations
Stop sending live observations	0x02	Command to stop sending live observations
Success	0x80	Response to indicate a command was successful
RFU	All other values	Reserved for Future Use

Table 3.35: GHS Control Point command and response Op Codes

3.5.2 GHS Control Point behavior

To start or stop sending live and temporarily stored observations during a connection via indications or notifications of the Live Health Observations characteristic, a client can write a command code to the GHS Control Point characteristic. Actual sending of such observations to the client will only happen when indications or notifications of the Live Health Observations characteristic have been enabled as well.

If the GHS Control Point has been enabled for indications, then the server shall respond with an indication of the GHS Control Point with a code indicating success of the command, even if the Live Health Observations characteristic has not been enabled for notifications or indications.

The server may be busy (e.g., with another procedure such as an RACP procedure), causing a delay in sending live observations after a successful start command.

After a successful stop command, the server shall not send any live or temporarily stored observations until after a subsequent successful start command.

If the GHS Control Point has not been enabled for indications, then the server responds on the write command with an ATT error code of 0xFD, Client Characteristic Configuration Descriptor Improperly Configured.

When a client writes a value to the GHS Control Point that does not match a supported command, then the server responds on the write command with an ATT error code of 0x81, Command Not Supported (see Table 2.1).

When a connection is established and indications or notifications of the Live Health Observations characteristic are enabled, then a GHS Server shall not send live or temporarily stored observations until the Start sending live observations command has been executed and the success response has been indicated.

3.6 Observation Schedule Changed

The Observation Schedule Changed characteristic shall only be present when the Service includes at least one Observation Schedule descriptor that can change. Observation schedule changes can be made by a client writing to an Observation Schedule descriptor or by a user via the user interface of the device or by some other means.

3.6.1 Observation Schedule Changed format

The format of the Observation Schedule Changed characteristic is defined in Table 3.36.

Field	Requirement	Data Type	Size (in octets)	Description
Observation Type	M	uint32	4	MDC code
Measurement Period	M	FLOAT	4	The measurement period in seconds. When the value is 0, the measurement period is unspecified.

Field	Requirement	Data Type	Size (in octets)	Description
Update Interval	M	FLOAT	4	The measurement interval in seconds. When the value is 0, the update interval is unspecified. This interval may be shorter than the Measurement Period if, for example, multiple averaging periods are overlapped to provide a rolling average.

Table 3.36: Observation Schedule Changed characteristic and Observation Schedule descriptor structure

3.6.2 Observation Schedule Changed behavior

When the Observation Schedule Changed characteristic is configured for indications, then the server shall inform connected clients of a change in an Observation Schedule descriptor by sending an indication of the Observation Schedule Changed characteristic with the same Observation Type, Measurement Period, and Update Interval value as the Observation Schedule descriptor.

Depending on the implementation, the server may or may not indicate such change to the client that caused the change by writing to the Observation Schedule descriptor.

3.7 Additional characteristic descriptors

There may be additional characteristic descriptors associated with the Health Sensor Features characteristic. These descriptors provide additional information for a single observation type. For each observation type supported by the server, each of these characteristic descriptors may be present. The following characteristic descriptor types are supported:

Characteristic Descriptor	Requirement	Mandatory Permissions	Optional Permissions	Security Permissions
Observation Schedule	O	Read	Write	None
Valid Range and Accuracy	O	Read	None	None

Table 3.37: Additional characteristic descriptors

3.7.1 Observation Schedule descriptor

An Observation Schedule descriptor may be used by a GHS Server to inform a client of the measurement period and update interval for a specific observation type supported by the server. The server reports the corresponding observation once every update interval and reports the observation calculated over the measurement period.

These descriptors apply to observations of a given type as identified by the Observation Type MDC code when reported via the Live Health Observations characteristic as well as when reported via the Stored Health Observations characteristic.

There shall be at most one Observation Schedule descriptor per supported observation type.

When the Observation Schedule descriptor is writable, then a client can update the schedule by writing to the descriptor.

3.7.1.1 Observation Schedule descriptor structure

The Observation Schedule descriptor has the same structure as the Observation Schedule Changed characteristic (see Table 3.36).

3.7.1.2 Observation Schedule descriptor behavior

A client that wants to know the observation schedules can read the Observation Schedule descriptors, when present.

When the descriptor supports writing, then a client can write to the descriptor to set the schedule for the specified observation type. If an Observation Schedule descriptor is written with a schedule that the server cannot support, then the write of the descriptor is rejected with a 0xFF – Out of Range error response. If an Observation Schedule Descriptor is written with a changed value of the Observation Type, then the write of the descriptor is rejected with a 0xFF – Out of Range error response.

When a server generates several types of observations normally at the same time, then a server should support an Observation Schedule descriptor for just one of these types of observations with a write property. For the other types of observations, a read-only Observation Schedule descriptor may or may not be supported.

3.7.2 Valid Range and Accuracy descriptor

A GHS Server can use a Valid Range and Accuracy descriptor to inform a client of the range and accuracy for a specific numeric, sampled observation type or numeric component supported by the server. There shall be at most one such descriptor for a given MDC code (i.e., there shall be at most one Valid Range and Accuracy descriptor per supported observation type or observation component).

The Valid Range and Accuracy descriptor supports the proper display of the observations in a pre-scaled graph on the UI of a client application.

3.7.2.1 Valid Range and Accuracy descriptor structure

The Valid Range and Accuracy descriptor has the following structure.

Field	Requirement	Data Type	Size (in octets)	Description
Observation Type	M	uint32	4	MDC code for a numeric, sampled observation or numeric component
Unit Code	M	uint16	2	A 2-byte MDC code from partition 2 that identifies the unit of the observation

Field	Requirement	Data Type	Size (in octets)	Description
Lower Limit	M	FLOAT	4	The lower limit of the observation the server supports in the unit given by the Unit code of this descriptor
Upper Limit	M	FLOAT	4	The upper limit of the observation the server supports in the unit given by the Unit code of this descriptor
Measurement Accuracy	O	FLOAT	4	The accuracy of the observation in the unit given by the Unit code of this descriptor

Table 3.38: Valid Range and Accuracy descriptor structure

3.7.3 Valid Range and Accuracy descriptor behavior

A client that wants to know the valid range and accuracy of the observations can read the Valid Range and Accuracy descriptors, when present.

This descriptor shall be static for the lifetime of a device.

4 SDP interoperability

Requirements in this section are defined as "Mandatory" (M), "Optional" (O), "Excluded" (X), “Not Applicable” (N/A), or "Conditional" (C.n). Conditional statements (C.n) are listed directly below the table in which they appear.

If this service is exposed over BR/EDR, then it shall have the following SDP record:

Item	Definition	Type	Value	Status
Service Class ID List	-	-	-	M
Service Class #0	-	UUID	«Generic Health Sensor»	M
Protocol Descriptor List	-	-	-	M
Protocol #0	-	UUID	L2CAP	M
Parameter #0 for Protocol #0	PSM	uint16	PSM = ATT	M
Protocol #1	-	UUID	ATT	M
BrowseGroupList	-	-	PublicBrowseRoot*	M

Table 4.1: SDP Record

* PublicBrowseRoot shall be present; however, other browse UUIDs may also be included in the list.

5 Acronyms and abbreviations

Acronym/Abbreviation	Meaning
ACOM	Abstract Content Information Model
ECG	Electrocardiogram
GHS	Generic Health Sensor (the combination of GHSP and GHSS)
GHSP	Generic Health Sensor Profile
GHSS	Generic Health Sensor Service
IEEE	Institute of Electrical and Electronics Engineers
MDC	Medical Device Communication
MTU	Maximum Transfer Unit
PDU	Protocol Data Unit
PHD	Personal Health Device
RACP	Record Access Control Point
RFU	Reserved for Future Use
TLV	Type-Length-Value
UDS	User Data Service

Table 5.1: Acronyms and abbreviations

6 References

[1] GATT Specification Supplement (GSS), v10 or later

[2] Bluetooth Assigned Numbers, https://www.bluetooth.com/specifications/assigned-numbers/

[3] Core Specification Supplement (CSS), v10 or later

[4] Bluetooth Core Specification, Version 4.2 or later

[5] Institute of Electrical and Electronics Engineers (IEEE), “11073-10101-2019 - IEEE Standard for Health informatics - Point-of-care medical device communication - Part 10101: Nomenclature”, October 2019, https://standards.ieee.org/standard/11073-10101-2019.html

[6] Institute of Electrical and Electronics Engineers (IEEE), “11073-10206 - Health informatics -- Device interoperability - Part 10206: Personal health device communication - Abstract information content model”, January 2023, http://standards.ieee.org/project/11073-10206.html

[7] Generic Health Sensor Profile Version 1.0 or later

[8] User Data Service, Version 1.1 or later

[9] Institute of Electrical and Electronics Engineers (IEEE), “11073-20601-2016 - ISO/IEC/IEEE International Standard - Health Informatics -- Personal Health Device Communication - Part 20601: Application Profile - Optimized Exchange Protocol”, June 2016, https://standards.ieee.org/standard/11073-20601-2016.html

[10] Elapsed Time Service, Version 1.0 or later

Appendix A Examples of Health Observation Bodies

A.1 Example 1 – Numeric – oxygen saturation

Field	Data Type	Size (in octets)	Example	Value (hex)	Little endian – transmission order
Observation Class Type	uint8	1	Numeric observation	01	01
Length	uint16	2	29 bytes	00 1D	1D 00
Flags	bits	2	0. Observation Type present - 1 1. Time Stamp present - 1 2. Measurement Duration present - 0 3. Measurement Status present - 0 4. Observation Id present - 0 5. Patient present - 0 6. Supplemental Information present - 1 7. Derived From present - 0 8. Is Member Of present - 0 9. TLVs present – 0 10. 10-15: RFU – 0 0b0000 0000 0010 0011	00 23	23 00
Observation Type	uint32	4	MDC_PULS_OXIM_SAT_O2 – 0x00024BB8	00 02 4B B8	B8 4B 02 00
Time Stamp	struct	9	Example from ETS [10]	22 00 00 29 2B 9D 72 00 06	22 72 9D 2B 29 00 00 06 00
Measurement Duration	FLOAT	4	absent	-	-
Measurement Status	struct	2	absent	-	-
Observation Id	uint32	4	absent	-	-
Patient	uint8	1	absent	-	-
Supplemental Information	struct	var	Count = 1 Codes = MDC_MODALITY_SPOT	01 00 02 4C 3C	01 3C 4C 02 00
Derived From	struct	var	absent	-	-
Is Member Of	struct	var	absent	-	-
TLVs	struct	var	absent	-	-
Observation Value	-				
Unit code	uint16	2	MDC_DIM_PER_CENT	02 20	20 02
Value	FLOAT	4	98	00 00 00 62	62 00 00 00

A.2 Example 2 - Compound observation – blood pressure

Field	Data Type	Size (in octets)	Example	Value (hex)	Little endian – transmission order
Observation Class Type	uint8	1	Compound observation	07	07
Length	uint16	2	53 bytes	00 35	35 00
Flags	bits	2	0. Observation Type present - 1 1. Time Stamp present - 1 2. Measurement Duration present - 0 3. Measurement Status present - 0 4. Observation Id present - 1 5. Patient present - 0 6. Supplemental Information present - 0 7. Derived From present - 0 8. Is Member Of present - 0 9. TLVs present – 0 10. 10-15: RFU – 0 0b0000 0000 0001 0011	00 13	13 00
Observation Type	uint32	4	MDC_PRESS_BLD_NONINV – 0x00024A04	00 02 4A 04	04 4A 02 00
Time Stamp	struct	9	Example from ETS [10]	22 00 00 29 2B 9D 72 00 06	22 72 9D 2B 29 00 00 06 00
Measurement Duration	FLOAT	4	absent	-	-
Measurement Status	struct	2	absent	-	-
Observation Id	uint32	4	for use in the next example in Section A.3 123456	00 01 E2 40	40 E2 01 00
Patient	uint8	1	absent	-	-
Supplemental Information	struct	var	absent	-	-
Derived From	struct	var	absent	-	-
Is Member Of	struct	var	absent	-	-
TLVs	struct	var	absent	-	-
Observation Value	-				
Number Of Components	uint8	1	3	03	03
Component 1	struct	13	component type = MDC_PRESS_BLD_NONINV_SYS	00 02 4A 05	05 4A 02 00
			component value type = 1 (numeric)	01	01
			unit code = MDC_DIM_MMHG	0F 20	20 0F
			value = 100	00 00 00 64	64 00 00 00
Component 2	struct	13	component type = MDC_PRESS_BLD_NONINV_DIA	00 02 4A 06	06 4A 02 00
			component value type = 1 (numeric)	01	01
			unit code = MDC_DIM_MMHG	0F 20	20 0F
			value = 60	00 00 00 3C	3C 00 00 00
Component 3	struct	13	component type = MDC_PRESS_BLD_NONINV_MEAN	00 02 4A 07	07 4A 02 00
			component value type = 1 (numeric)	01	01
			mmHG = MDC_DIM_MMHG	0F 20	20 0F
			value = 80	00 00 00 50	50 00 00 00

A.3 Example 3 – Compound discrete – blood pressure status

Field	Data Type	Size (in octets)	Example	Value (hex)	Little endian – transmission order
Observation Class Type	uint8	1	Compound discrete observation	05	05
Length	uint16	2	23 bytes	00 17	17 00
Flags	bits	2	0. Observation Type present - 1 1. Time Stamp present - 0 2. Measurement Duration present - 0 3. Measurement Status present - 0 4. Observation Id present - 0 5. Patient present - 0 6. Supplemental Information present - 0 7. Derived From present - 1 8. Is Member Of present - 0 9. TLVs present – 0 10. 10-15: RFU – 0 0b0000 0000 1000 0001	00 81	81 00
Observation Type	uint32	4	MDC_BLP_MEASUREMENT STATUS - 0x008055F0	00 80 55 F0	F0 55 80 00
Time Stamp	struct	9	absent	-	-
Measurement Duration	FLOAT	4	absent	-	-
Measurement Status	struct	2	absent	-	-
Observation Id	uint32	4	absent	-	-
Patient	uint8	1	absent	-	-
Supplemental Information	struct	var	absent	-	-
Derived From	struct	var	1 item 123456 – from earlier example in Section A.2	01 00 01 E2 40	01 40 E2 01 00
Is Member Of	struct	var	absent	-	-
TLVs	struct	var	absent	-	-
Observation Value	-				
Number Of Terms	uint8	1	2	02	02
Term 1	uint32	4	Cuff loose (3::240)	00 03 00 F0	F0 00 03 00
Term 2	uint32	4	Cuff improperly placed (3::430)	00 03 01 AE	AE 01 03 00

A.4 Example 4 – Sample array – ECG waveform

Field	Data Type	Size (in octets)	Example	Value (hex)	Little endian – transmission order
Observation Class Type	uint8	1	Sample array observation	04	04
Length	uint16	2	151 bytes	00 97	97 00
Flags	bits	2	0. Observation Type present - 1 1. Time Stamp present - 1 2. Measurement Duration present - 0 3. Measurement Status present - 0 4. Observation Id present - 0 5. Patient present - 0 6. Supplemental Information present - 0 7. Derived From present - 0 8. Is Member Of present - 0 9. TLVs present – 0 10. 10-15: RFU – 0 0b0000 0000 0000 0011	00 03	03 00
Observation Type	uint32	4	MDC_ECG_ELEC_POTL_I – 0x00020101	00 02 01 01	01 01 02 00
Time Stamp	struct	9	Example from ETS [10]	22 00 00 29 2B 9D 72 00 06	22 72 9D 2B 29 00 00 06 00
Measurement Duration	FLOAT	4	absent	-	-
Measurement Status	struct	2	absent	-	-
Observation Id	uint32	4	absent	-	-
Patient	uint8	1	absent	-	-
Supplemental Information	struct	var	absent	-	-
Derived-From	struct	var	absent	-	-
Is Member Of	struct	var	absent	-	-
TLVs	struct	var	absent	-	-
Observation Value	-				
Unit Code	uint16	2	MDC_DIM_MILLI_VOLT	10 B2	B2 10
Scale Factor	FLOAT	4	1.612	00 00 06 4C	4C 06 00 00
Offset	FLOAT	4	2048	00 00 08 00	00 08 00 00
Sample Period	FLOAT	4	0.010 (10 msec)	FD 00 00 0A	0A 00 00 FD
Number Of Samples Per Period	uint8	1	1	01	01
Bytes Per Sample	uint8	2	2	02	02
Number Of Samples	uint32	4	116	74	74
Scaled Samples	variable	232	2041 2043 2037 2047 2060 2062 2051 2023 2014 2027 2034 2033 2040 2047 2047 2053 2058 2064 2059 2063 2061 2052 2053 2038 1966 1885 1884 2009 2129 2166 2137 2102 2086 2077 2067 2067 2060 2059 2062 2062 2060 2057	07F9 07FB 07F5 07FF 080C 080E 0803 07E7 07DE 07EB 07F2 07F1	F9 07 FB 07 F5 07 FF 07 0C 08 0E 08 03 08 E7 07 DE 07 EB 07 F2 07 F1 07 F8 07 FF 07 FF 07

Field	Data Type	Size (in octets)	Example	Value (hex)	Little endian – transmission order
			2045 2047 2057 2054 2042 2029 2027 2018 2007 1995 2001 2012 2024 2039 2068 2092 2111 2125 2131 2148 2137 2138 2128 2128 2115 2099 2097 2096 2101 2101 2091 2073 2076 2077 2084 2081 2088 2092 2070 2069 2074 2077 2075 2068 2064 2060 2062 2074 2075 2074 2075 2063 2058 2058 2064 2064 2070 2074 2067 2060 2062 2063 2061 2059 2048 2052 2049 2048 2051 2059 2059 2066 2077 2073	07F8 07FF 07FF 0805 080A 0810 080B 080F 080D 0804 0805 07F6 07AE 075D 075C 07D9 0851 0876 0859 0836 0826 081D 0813 0813 080C 080B 080E 080E 080C 0809 07FD 07FF 0809 0806 07FA 07ED 07EB 07E2 07D7 07CB 07D1 07DC 07E8 07F7 0814 082C 083F 084D 0853 0864 0859 085A 0850 0850 0843 0833 0831 0830 0835 0835 082B 0819 081C 081D 0824 0821 0828 082C 0816 0815 081A 081D 081B 0814 0810 080C 080E 081A 081B 081A 081B 080F 080A 080A 0810 0810 0816 081A 0813 080C 080E 080F 080D 080B 0800 0804 0801 0800 0803 080B 080B 0812 081D 0819	05 08 0A 08 10 08 0B 08 0F 08 0D 08 04 08 05 08 F6 07 AE 07 5D 07 5C 07 D9 07 51 08 76 08 59 08 36 08 26 08 1D 08 13 08 13 08 0C 08 0B 08 0E 08 0E 08 0C 08 09 08 FD 07 FF 07 09 08 06 08 FA 07 ED 07 EB 07 E2 07 D7 07 CB 07 D1 07 DC 07 E8 07 F7 07 14 08 2C 08 3F 08 4D 08 53 08 64 08 59 08 5A 08 50 08 50 08 43 08 33 08 31 08 30 08 35 08 35 08 2B 08 19 08 1C 08 1D 08 24 08 21 08 28 08 2C 08 16 08 15 08 1A 08 1D 08 1B 08 14 08 10 08 0C 08 0E 08 1A 08 1B 08 1A 08 1B 08 0F 08 0A 08 0A 08 10 08 10 08 16 08 1A 08 13 08 0C 08 0E 08 0F 08 0D 08 0B 08 00 08 04 08 01 08 00 08 03 08 0B 08 0B 08 12 08 1D 08 19 08

A.5 Example 5 – Observation bundle

Field	Data Type	Size (in octets)	Example	Value (hex)	Little endian – transmission order
Observation Class Type	uint8	1	Observation bundle	FF	FF
Length	uint16	2	53 bytes	00 35	35 00
Flags	bits	2	0. Observation Type present - 0 1. Time Stamp present - 1 2. Measurement Duration present - 0 3. Measurement Status present - 0 4. Observation Id present - 0 5. Patient present - 0 6. Supplemental Information present - 1 7. Derived From present - 0 8. Is Member Of present - 0 9. TLVs present – 0 10. 10-15: RFU – 0 0b0000 0000 0100 0010	00 42	42 00
Observation Type	uint32	4	absent	-	-
Time Stamp	struct	9	Example from ETS [10]	22 00 00 29 2B 9D 72 00 06	22 72 9D 2B 29 00 00 06 00
Measurement Duration	FLOAT	4	absent	-	-
Measurement Status	struct	2	absent	-	-
Observation Id	uint32	4	absent	-	-
Patient	uint8	1	absent	-	-
Supplemental Information	struct	var	Count = 1 Codes = MDC_MODALITY_SPOT	01 00 02 4C 3C	01 3C 4C 02 00
Derived From	struct	var	absent	-	-
Is Member Of	struct	var	absent	-	-
TLVs	struct	var	absent	-	-
Observation Value	-				
Number Of Observations	uint8	1	2	02	02
Bundled Observation 1	struct	15	See Section A.5.1	-	01 0F 00 01 00 B8 4B 02 00 20 02 62 00 00 00
Bundled Observation 2	struct	15	See Section A.5.2	-	01 0F 00 01 00 1A 48 02 00 A0 0A 62 00 00 00

A.5.1 Bundled observation 1 – SpO2

Field	Data Type	Size (in octets)	Example	Value (hex)	Little endian – transmission order
Observation Class Type	uint8	1	Numeric observation	01	01
Length	uint16	2	15 bytes	00 0F	0F 00
Flags	bits	2	0. Observation Type present - 1 1. Time Stamp present - 0 2. Measurement Duration present - 0	00 01	01 00

Field	Data Type	Size (in octets)	Example	Value (hex)	Little endian – transmission order
			3. Measurement Status present - 0 4. Observation Id present - 0 5. Patient present - 0 6. Supplemental Information present - 0 7. Derived From present - 0 8. Is Member Of present - 0 9. TLVs present – 0 10. 10-15: RFU – 0 0b0000 0000 0000 0001		
Observation Type	uint32	4	MDC_PULS_OXIM_SAT_O2 – 0x00024BB8	00 02 4B B8	B8 4B 02 00
Observation Value	-				
Unit Code	uint32	4	MDC_DIM_PER_CENT	02 20	20 02
Value	FLOAT	4	98	00 00 00 62	62 00 00 00

A.5.2 Bundled observation 2 – Heart Rate

Field	Data Type	Size (in octets)	Example	Value (hex)	Little endian – transmission order
Observation Class Type	uint8	1	Numeric observation	01	01
Length	uint16	2	15 bytes	00 0F	0F 00
Flags	bits	2	0. Observation Type present - 1 1. Time Stamp present - 0 2. Measurement Duration present - 0 3. Measurement Status present - 0 4. Observation Id present - 0 5. Patient present - 0 6. Supplemental Information present - 0 7. Derived From present - 0 8. Is Member Of present - 0 9. TLVs present – 0 10. 10-15: RFU – 0 0b0000 0000 0000 0001	00 01	01 00
Observation Type	uint32	4	MDC_PULS_OXIM_PULS_RATE – 0x0002481A	00 02 48 1A	1A 48 02 00
Observation Value	-				
Unit Code	uint32	4	MDC_DIM_BEAT_PER_MIN	0A A0	A0 0A
Value	FLOAT	4	98	00 00 00 62	62 00 00 00

Appendix B Examples of RACP usage

The examples in the tables below show how RACP is used in GHSS.

B.1 Counting records

Counting records is done by using the Report number of stored records Op Code in a write to the RACP as shown below.

Request	Encoded format
Report number of stored records – all	{0x04, 0x01}
Report number of stored records ≥ record number 0xAABBCCDD	{0x04, 0x03, 0x01, 0xDD, 0xCC, 0xBB, 0xAA}
Report number of stored records ≥ time value 0xAABBCCDDEEFF (if operand is supported)	{0x04, 0x03, 0x02, 0xFF, 0xEE, 0xDD, 0xCC, 0xBB, 0xAA}

Table B.1: RACP Counting requests

The response is an indication of the RACP with either a Number of stored records response when records could be counted, including zero records, or a Response code with a specific error code.

Responses	Encoded format
Number of stored records – 0xAABBCCDD records found	{0x05, 0x00, 0xDD, 0xCC, 0xBB, 0xAA}
Number of stored records – 0 records found	{0x05, 0x00, 0x00, 0x00, 0x00, 0x00}
Response code – Server busy	{0x06, 0x00, 0x04, 0x0A}
Response code – operand not supported	{0x06, 0x00, 0x04, 0x09}

Table B.2: RACP Counting responses

B.2 Retrieving records

After a client writes a command to the RACP with the Combined report Op Code, the server notifies all records on the Stored Health Observations characteristic and indicates the number found afterwards on the RACP.

Request	Encoded format
Combined report of records – all	{0x07, 0x01}
Combined report of records ≥ record number 0xAABBCCDD	{0x07, 0x03, 0x01, 0xDD, 0xCC, 0xBB, 0xAA}
Combined report of records ≥ time value 0xAABBCCDDEEFF (if operand is supported)	{0x07, 0x03, 0x02, 0xFF, 0xEE, 0xDD, 0xCC, 0xBB, 0xAA}

Request	Encoded format
Combined report of records – first (if operator is supported)	{0x07, 0x05}
Combined report of records – last (if operator is supported)	{0x07, 0x06}

Table B.3: RACP Combined report requests

Successful retrieval of some records results in a Combined report response indication. When no records are found or an error condition occurs, then a response code with an error code is indicated.

Responses	Encoded format
Combined report response – 0xAABBCCDD records found	{0x08, 0x00, 0xDD, 0xCC, 0xBB, 0xAA}
Response code – No records found	{0x06, 0x00, 0x07, 0x06}
Response code – Server busy	{0x06, 0x00, 0x07, 0x0A}
Response code – Operand not supported	{0x06, 0x00, 0x07, 0x09}
Response code – Operator not supported	{0x06, 0x00, 0x07, 0x04}

Table B.4: RACP Combined report responses

B.3 Deleting records

The server may optionally support the deletion of records.

Request	Encoded format
Delete stored records – all	{0x02, 0x01}

Table B.5: RACP Delete stored record report request

Successful deletion of some records results in a Response code indication of success. Otherwise, a Response code with an error code is indicated.

Responses	Encoded format
Response code – Success	{0x06, 0x00, 0x02, 0x01}
Response code – No records found	{0x06, 0x00, 0x02, 0x06}
Response code – Server busy	{0x06, 0x00, 0x02, 0x0A}
Response code – Op Code not supported	{0x06, 0x00, 0x02, 0x02}

Table B.6: RACP Delete stored record report responses