RW BLE Environmental Sensing Profile Interface Specification

Interface Specification

RW-BLE-ENVP-IS

Version 2

2016-05-05

RivieraWaves
http://www.rivierawaves.com/

Revision History

Version	Date	Revision Description	Author
0.1	February 23th 2016	Initial draft	VF
1.0	April 20th 2016	New draft	GF
2.0	May 5th 2016	Update following review	GF

Table of Contents

1 Overview

1.1 Document Overview

This document describes the non-standard interface of the RivieraWaves (RW) Bluetooth Low Energy (BLE) Environmental Sensing Profile (ENVP) implementation. Along this document, the interface messages will be referred to as API messages for the profile block(s).

Their description will include their utility and reason for implementation for a better understanding of the user and the developer that may one day need to interface them from a higher application.

1.2 BLE Environmental Sensing Profile Overview

The ENVP enables a collector device to connect and interact with Environmental Sensing Device sharing its sensor's data.

This service has been implemented as a profile. Within this profile, two roles can be supported: Sensor role (ENVS) and Collector role (ENVC). The Collector role must support the GAP Central Role and the Sensor role, the GAP Peripheral role. The profile requires a connection to be established between the two devices for its functionality.

The various documents edited by the Bluetooth SIG present different use cases for this profile, their GATT, GAP and security, mandatory and optional requirements. The Environmental Sensing Profile specifications have been adopted by the Bluetooth SIG on November 18th 2014 (**Error! Reference source not found.** and **Error! Reference source not found.**). Their related Test Specifications have been released and are referenced in **Error! Reference source not found.** and **Error! Reference source not found.**.

The profile is implemented in the RW-BLE software stack as two tasks, one for each role. Each task has an API decided after the study of the profile specifications and test specifications, and it is considered to be minimalistic and designed for a future application which would combine the profile functionality with the device connectivity and security procedures.

1.3 Environmental Sensing Characteristics

The Environmental Sensing service supports a large number of different sensor types. Each sensor type has a different characteristic dedicated to it consisting of the following set of characteristics shown below in the Table 1.

Characteristic Name	Characteristic Enumerator	Properties
Descriptor Value Changed	**ENVP_ES_DESCRIPTOR_VALUE_CHANGED_CHAR**	Indicate
Apparent Wind Direction	**ENVP_ES_APRNT_WIND_DIRECTION_CHAR**	Read/Notify
Apparent Wind Speed	**ENVP_ES_APRNT_WIND_SPEED_CHAR**	Read/Notify
Dew Point	**ENVP_ES_DEW_POINT_CHAR**	Read/Notify
Elevation	**ENVP_ES_ELEVATION_CHAR**	Read/Notify
Gust Factor	**ENVP_ES_GUST_FACTOR_CHAR**	Read/Notify
Heat Index	**ENVP_ES_HEAT_INDEX_CHAR**	Read/Notify

Humidity	**ENVP_ES_HUMIDITY_CHAR**	Read/Notify
Irradiance	**ENVP_ES_IRRADIANCE_CHAR**	Read/Notify
Pollen Concentration	**ENVP_ES_POLLEN_CONC_CHAR**	Read/Notify
Rainfall	**ENVP_ES_RAINFALL_CHAR**	Read/Notify
Pressure	**ENVP_ES_PRESSURE_CHAR**	Read/Notify
Temperature	**ENVP_ES_TEMPERATURE_CHAR**	Read/Notify
True Wind Direction	**ENVP_ES_TRUE_WIND_DIR_CHAR**	Read/Notify
True Wind Speed	**ENVP_ES_TRUE_WIND_SPEED_CHAR**	Read/Notify
UV Index	**ENVP_ES_UV_INDEX_CHAR**	Read/Notify
Wind Chill	**ENVP_ES_WIND_CHILL_CHAR**	Read/Notify
Barometric Pressure Trend	**ENVP_ES_BAR_PRES_TREND_CHAR**	Read/Notify
Magnetic Declination	**ENVP_ES_MAGN_DECLINE_CHAR**	Read/Notify
Magnetic Flux Density - 2D	**ENVP_ES_MAGN_FLUX_2D_CHAR**	Read/Notify
Magnetic Flux Density - 3D	**ENVP_ES_MAGN_FLUX_3D_CHAR**	Read/Notify

Table 1 – Enumerated list of ES Characteristics - *enum envp_es_char*

All Characteristics except the “Descriptor Value Changed” can support the set of descriptors given in Table 2

Descriptor Name	Requirements	Properties	Security
ES Client Char Configuration	Mandatory if notifiy is to be supported.	Read/Write	None
ES Measurement	Mandatory if Multiple instances of characteristic exist, otherwise optional	Read	None
ES Trigger Setting	Mandatory if Notify supported	Read/Write/ Notify	Bonding for Write
ES Trigger Setting	Optional	Read/Write/ Notify	Bonding for Write
ES Trigger Setting	Optional	Read/Write/ Notify	Bonding for Write
ES Configuration	Mandatory If More than one ES Trigger Setting Exist	Read/Write	Bonding for Write
Characteristic User Description	Optional	Read/Write	Bonding for Write
Valid Range	Optional	Read	None
Extended Properties	Optional, Required to allow client write User Description.	Read	None

Table 2 –list of ES Descriptors

Any number and combination of ES measurement characteristics are supported by the profile. More than one instance of each measurement characteristic is allowed, with each instance being distinguished by its Measurement descriptor.

Similarly as most of the descriptors are optional, the system can be configured to support different descriptors and numbers of triggers.

1.4 Multiple Characteristic Instance

To uniquely identify each measurement characteristic both an **es_char_id** and **es_char_inst** are used. **The es_char_id** is taken form the list if enumerators shown in Table 1, while the **es_char_inst** is based on the order of discovery in the client or creation in the server. In the client a unique **es_char_inst** is assigned for each instance of a discovered characteristic type.

These two parameters are used extensively on the Profile API interfaces in both the client and server. If multiple instances of a given characteristic type do not exist then the **es_char_inst** will always be the default value (= 0).

When multiple instances of a characteristic are available – a separate measurement descriptor will be available for each instance. This information determines how they differ, in obtaining sensor information. (for example if a two temperate characteristics are exposed – the measurement descriptor may indicate they have different application -via the Application Field – for example one is for soil temperature, the other for Air Temperature).

The format of the measurement descriptor is shown below in Table 3

Field Name	Type	Purpose
flags	**uint16_t**	Currently not used
samp_func	**uint8_t**	Identifies mechanism used by sensor for sampling (values 0-7) see [1] for more information
meas_period	**uint32_t (only 24 bits valid)**	The time over which the measurement is taken
update_period	**uint32_t (only 24 bits valid)**	Frequency at which the server updates the measurement
App	**uint8_t**	Identifies the application area of the sensor (e.g soil or air) from one of 33 pre-defined app types detailed in [1]
meas_uncertainty	**uint8_t**	The measurement uncertainty expressed as a percentage

Table 3 - Format of the Measurement Descriptor (struct envp_es_meas_desc)

1.5 Triggers and Trigger Configuration

Triggers are used on the Server to limit the notification of changes to the client to specific conditions. They are used in combination with the CCC descriptor to configuration the conditions on which notifications are sent to the client.

The profile supports upto 3 separate triggers conditions (identified using *'**triggerIdx**'* field of messages) which can be logically combined to further control the conditions for notification. The configuration descriptor controls the logic operation of the triggers, and is required if more than one trigger is supported. If only a single trigger is supported, then the configuration descriptor is not required.

Due to the close coupling between measurements and triggers, the functionality of the triggers is located outside of the profile, in the user application. This removes the overhead of the application reporting every measured value to the profile (if the trigger functionality was located in the profiles). Instead, the profile provides an interface to access and modify the triggers in the application.

The Client Configuration Characteristic (CCC) is very closely coupled with the Triggers and thus also located outside the profile in the application. In fact the CCC is an integral part of the trigger mechanism.

Figure 1, shows the relationship between triggers and CCC. When the sensor obtains a new measurement, the application determines if the new measurement meets all/or any of the trigger conditions. The requirement to meet all or any of the trigger conditions is determined by the current setting of the trigger configuration. If this is satisfied the application stores a new ES Sensor value. This new value can be read by the peer device, and if the Client Configuration Characteristic (CCC) is enabled the client can be notified by the application of the new value.

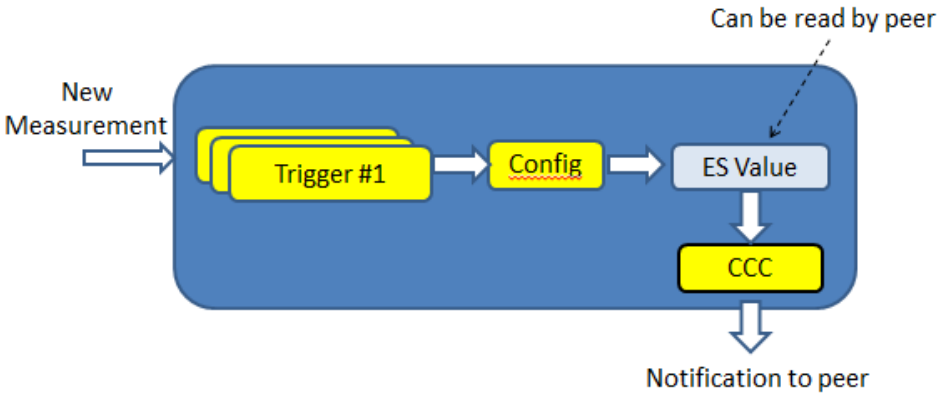

Figure 1 – Triggers and the CCC

The triggers and trigger configuration are specific for each client. Thus multiple sets of triggers and trigger configurations are maintained in the server application – one for each active link. The trigger information can also persist across multiple links – for bonded devices – so the server application has to maintain trigger information for bonded devices in between connections.

The parameters used to describe a Trigger on the API is shown in Table 4 :

Field Name	Type	Purpose
condition	**uint8_t**	The trigger condition, values 0-9. The interpretation of these values is given in Table 5
time_based_trigger	**uint32_t** **(only 24 bits valid)**	If trigger condition is set to 2 or 3. Then time based triggers apply. Otherwise this parameter should be zero.
meas_value_trigger	**union envp_trigger_val_char**	This is a union of measurement values, with the different measurement type dependent on the current sensor type (selected by the es_char_id). This field only applies when condition has a value from 4-9, otherwise this field has a value = 0. For some es_char_ids this field is not applicable. (eg. ENVP_ES_MAGN_FLUX_2D_CHAR and ENVP_ES_MAGN_FLUX_3D_CHAR)
triggerIdx	**uint8_t**	This is used to identifier which of 3 triggers the command applies to. The valid values for this field are

		0,1,2. If only a single trigger exists, then this field will always be 0.

Table 4 :- API parameters used to describe a trigger

Trigger Condition	Value	Description
ENVC_TRIGGER_INACTIVE	0	Trigger Inactive
ENVC_TRIGGER_FIXED_TIME_INT	1	Use a fixed time interval between transmissions
ENVC_TRIGGER_NO_LESS_THAN_TIME	2	No less than the specified time between transmissions
ENVC_TRIGGER_WHEN_VALUE_CHANGES	3	When value changes compared to previous value
ENVC_TRIGGER_WHILE_LESS_THAN	4	While less than the specified value
ENVC_TRIGGER_WHILE_LESS_THAN_OR_EQUAL	5	While less than or equal to the specified value
ENVC_TRIGGER_WHILE_GREATER_THAN	6	While greater than the specified value
ENVC_TRIGGER_WHILE_GREATER_THAN_OR_EQUAL	7	While greater than or equal to the specified value
ENVC_TRIGGER_WHILE_EQUAL_TO	8	While equal to the specified value
ENVC_TRIGGER_WHILE_NOT_EQUAL_TO	9	While not equal to the specified value
ENVC_TRIGGER_RESERVED_VALUE	10	Reserved for future use

Note :- Some trigger conditions are not possible for some ES characteristics as described in [1]

Table 5 :- Trigger Conditions

1.6 Multiple Trigger Configuration

If multiple triggers are supported then the Configuration Descriptor is required to determine how to combine the triggers. The configuration descriptor contains a trigger logic value. This is represented on the API by the "***trigger_logic***" field which has the values shown in Table 6.

Value	Description
0	BOOLEAN AND of two or more triggers
1	BOOLEAN OR of two or more triggers

Table 6 :- Values of the trigger_logic field on the API

1.7 Descriptor Value Changed (DVC)

This characteristic is used to inform the client(s) if any of the Descriptors for any of the ES Measurement Characteristics have changed on the server. It is mandatory if any of the following descriptors can be changed by the Server: Measurement, Trigger Setting, Configuration, or Characteristic User Description. It is also mandatory if the Server supports the write property for the Characteristic User Description descriptor.

It enables a Server to alert bonded Clients that the value of one or more descriptors for a given Characteristic have changed and need to be re-read.

The "ENVC_DVC_IND" event provides the Client Application with the identifier of the Characteristic whose descriptors have changed, and an indication of the descriptors which have changed (contained in a 'flags' field). This allows the Client to re-read the values of these descriptors. However, if more than one instance of the ES characteristic exists – it does not provide any information on the instance which has changed. Thus to determine the new descriptor value the application will have to read the descriptor for each instance of the characteristic which was indicated as updated.

Similarly, if a trigger has been changed by the server – and the characteristic supports multiple triggers – the '**flags**' field does not indicate which of the triggers had changed. Thus the application will have to re-read each of the triggers in-turn to determine which one has changed.

Bit (of flags)	
0	Source of Change (0 – Clent, 1 – Server)
1	Trigger Setting Descriptor modified (0 – False, 1 – True)
2	Configuration Descriptor modified (0 – False, 1 True)
3	Measurement Descriptor modified (0 – False, 1 True)
4	User Description Descriptor modified (0 – False, 1 True)
5-16	Reserved

Table 7 - Format of the Flags Field.

1.8 Valid Ranges

The server maintains a valid range for each different ES measurement type. The formats and resolution are dependent on the ES measurement type. The valid range can be updated by the server and read from the client. To support API messages related to the Valid Range, the API exchanges valid range information using a union of range values called ***envp_range_char***, with the relevant field of the union being selected based on the ***es_char_id*** carried in the same message. Each field of the union consists of a structure which contains *lower* and *upper* fields.

Table 8 and Table 9 overview the types of the valid ranges as defined in [1]. While Table 10 details the format of the union ***envp_range_char*** used on the API to transfer range information.

ES Characteristic	Type	Unit	Resolution
Apparent Wind Direction	Uint16	Degrees	0,01 Degrees
Apparent Wind Speed	Uint16	Unit is in meters per second	0.01 m/s
Dew Point	Sint8	degrees celsius	1 degree Celsius
Elevation	Sint24	Meters	0.01 m
Gust Factor	Uint8	unitless	0.1
Heat Index	Sint8	degrees celsius	1 degree Celsius
Humidity	Uint16	Percentage	0.01 percent
Irradiance	Uint16	Unit is in watt per square meter	0.1 W/m^2
Pollen Concentration	Uint24	concentration per cubic meter	1/m^3
Rainfall	Uint16	Meters	1 mm
Pressure	Uint32	Pascals	0.1 Pa
Temperature	Sint16	degrees celsius	0.01 degree Celsius
True Wind Direction	Uint16	Degrees	0.01 degrees
True Wind Speed	Uint16	meters per second	0.01 m/s
UV_Index	Uint8	Unitless	1
Wind Chill	Sint8	degrees celsius	1 degree Celsius
Barometric Pressure Trend	Uint8	Enumerated format (See Note 1)	
Magnetic Declination	Uint16	degrees	0.01 degrees
Magnetic Flux Density 2D	Sint16	Tesla	1 x 10^-7 Tesla
Magnetic Flux Density 3D	Sint16	Tesla	1 x 10^-7 Tesla

Table 8:- Format of the Valid Range per Characteristic

Key	Value
0	Unknown
1	Continuously falling
2	Continuously rising
3	Falling, then steady
4	Rising, then steady
5	Falling before a lesser rise
6	Falling before a greater rise
7	Rising before a greater fall
8	Rising before a lesser fall
9	Steady
10 – 255	Reserved for future use

Table 9 :- Format of Barometric Pressure Trend – Valid Range

Field Name	Type	Sub-Fields
range_aprnt_wind_dir	*range_uint16_t*	*uint16_t lower,* *uint16_t upper*
range_aprnt_wind_speed	*range_uint16_t*	*uint16_t lower,* *uint16_t upper*
range_dew_point	*range_int8_t*	*int8_t lower,* *int8_t upper*
range_elevation	*range_int24_t*	*int32_t lower, (note 1)* *int32_t upper*
range_gust_factor	*range_uint8_t*	*uint8_t lower,* *uint8_t upper*
range_heat_index	*range_int8_t*	*int8_t lower,* *int8_t upper*
range_humidity	*range_uint16_t*	*uint16_t lower,* *uint16_t upper*
range_Irradiance	*range_uint16_t*	*uint16_t lower,* *uint16_t upper*
range_pollen_conc	*range_uint24_t*	*uint32_t lower, (note 1)* *uint32_t upper*
range_rainfall	*range_uint16_t*	*uint16_t lower,* *uint16_t upper*
range_pressure	*range_uint32_t*	*uint32_t lower,* *uint32_t upper*
range_temperature	*range_int16_t*	*int16_t lower,* *int16_t upper*
range_true_wind_dir	*range_uint16_t*	*uint16_t lower,* *uint16_t upper*
range_true_wind_speed	*range_uint16_t*	*uint16_t lower,* *uint16_t upper*
range_uv_index	*range_uint8_t*	*uint8_t lower,* *uint8_t upper*
range_wind_chill	*range_int8_t*	*int8_t lower,* *int8_t upper*
range_bar_pres_trend	*range_uint8_t*	*uint8_t lower,* *uint8_t upper*
range_magn_decline	*range_uint16_t*	*uint16_t lower,* *uint16_t upper*
range_flux_2d	*range_int16_t*	*int16_t lower,* *int16_t upper*
range_flux_3d	*range_int16_t*	*int16_t lower,* *int16_t upper*

Note 1 – uint24 and int24 values are carried over the API as uint32_t and int32_t values. The range of these values is restricted to uint24/int24 values, and in the case of int24 the sign is preserved.

Table 10 :- Fields of the *envp_range_char* union

1.9 ES Value

The ES Value refers to the value of given sensor characteristic. It can be read directly from the client or if notifications are enabled it will be indicated to the client each time it changes value. The format of the ES Value is dependent on the type of Sensor. Thus in all API messages the interpretation of the ES Value is dependent on the value of "***es_char_id***". Table 11 below describes the format of the union (***envp_val_char***) used to carry the ES Values in API messages.

Field Name	Type
apparent_wind_direction	*uint16_t*
apparent_wind_speed	*uint16_t*
dew_point	*int8_t*
elevation	*int32_t (note 1)*
gust_factor	*uint8_t*
heat_index	*int8_t*
humidity	*uint16_t*
irradiance	*uint16_t*
pollen_concentration	*uint32_t (note 1)*
rainfall	*uint16_t*
pressure	*uint32_t*
temperature	*int16_t*
true_wind_direction	*uint16_t*
true_wind_speed	*uint16_t*
uv_index	*uint8_t*
wind_chill	*int8_t*
barometric_pressure_trend	*uint8_t*
magnetic_declination	*uint16_t*
mag_flux_dens_2D	*magnetic_flux_dens_2D*
mag_flux_dens_3D	*magnetic_flux_dens_3D*

Note 1 – uint24 and int24 values are carried over the API as uint32_t and int32_t values. The range of these values is restricted to uint24/int24 values, and in the case of int24 the sign is preserved.

Table 11 :- Fields of union *envp_val_char*, used to represent ES Values.

The *magnetic_flux_dens_2D* and *magnetic_flux_dens_3D* are two specific structures used to represent the flux density values. Their format is shown in Table 12 and 13

Field Name	Type
x_axis	*int16_t*
y_axis	*int16_t*

Table 12 :- Fields of structure *magnetic_flux_dens_2D*, used to represent 2D flux density ES Values.

Field Name	Type
x_axis	*int16_t*
y_axis	*int16_t*
z_axis	*int16_t*

Table 13 :- Fields of structure *magnetic_flux_dens_3D*, used to represent 3D flux density ES Values.

1.10 Client Discovery

The client discovery consists of using the GATT to discover the Service, Characteristics and Descriptors available in the Envp profile server. The discovered information is presented to the client to allow it to cache it and restore for subsequent connections to the same server. The application is informed of each environmental sensing characteristic (and the Descriptor Value changed Characteristic) via the ENVC_ES_INFO_IND message. This carries a complete description of the characteristic, including its handles and properties. The handles of all the descriptors associated with the characteristic are also included

This information is used by the application to restore previously discovered information for a given server. The information in this is **not intended to be used in normal operation of the client to read/write elements in the server**. Thus there is no need for the application to interpret most of the information presented in this event. This information normally abstracted from the application in the other messages on the API.

The fields of the ENVC_ES_INFO_IND message are shown and described in Table 14, this maps to the structure *envc_es_char_info*. Many of the elements are optional, if they are not present in the server – their absence is indicated by a Zero value (INVALID_HANDLE).

Field	Description	Mandatory Optional
es_char_id	The identifier for the ES characteristic see Table 1	M
es_char_inst	The instance of the ES characteristic. If only one instant is present this value is always 0.	M
es_char_hdl	The attribute handle of the characteristic declaration	M
es_val_hdl	The attribute handle of the characteristic value	M
es_end_hdl	The end handle for the characteristic declaration	M
prop	The characteristic properties field.	M
ccc_desc_hdl	The handle of the client configuration characteristic descriptor	O
meas_desc_hdl	The handle of the measurement description descriptor	O
trigger_1_hdl	The handle of the first trigger descriptor	O
trigger_2_hdl	The handle of the second trigger descriptor	O
trigger_3_hdl	The handle of the third trigger descriptor	O
config_desc_hdl	The handle of the configuration settings descriptor	O
user_description_hdl	The handle of the user description descriptor	O
range_desc_hdl	The handle of the valid range descriptor	O
ext_prop_desc_hdl	The handle of the extended properties descriptor	O

Table 14 :- Format of *envc_es_char_info* structure

1.11 Parition of Storage

In the profile different assumptions are made about the storage of Characteristic Values and Descriptors. The Characteristic Values are stored by the application with the storage of the Characteristic descriptors being partitioned between the ATT DB and the Application. The decision to allocate a descriptor to the ATT_DB or the Application is based on the "dynamic" nature of the descriptor. Descriptors which are infrequently changed can be stored in the ATT_DB, however descriptors which are more dynamic in nature (ie. Triggers and Config) are allocated to the application for storage.

Name	Location	Security
Characteristic Value	Application	None
ES Client Char Configuration	Application	None
ES Measurement	ATT_DB	None
ES Trigger Setting	Application	Bonding for Write
ES Trigger Setting	Application	Bonding for Write
ES Trigger Setting	Application	Bonding for Write
ES Configuration	Application	Bonding for Write
Characteristic User Description	Application	Bonding for Write
Valid Range	ATT_DB	None
Extended Properties	ATT_DB	None

Table 15 :- Partitioning of Descriptors between ATT_DB and Application

2 ENVP Sensor Role API

2.1 Environment

This Implementation has no states and made transparent for reading.

2.2 Implementation

The Server handles requests from Application [APP] and Client for Reading/Writing Characteristics attributes/ descriptors and values

2.2.1 Modification of Descriptors from the APP:

The application is required to set the configuration of Characteristics for the profile during initialization stage. The chosen set of characteristics should also contain the attribute descriptors and write configurations. The configuration is set once and it is not possible to modify it during run time of the server.

The Application can modify any ES measurement value, perform the trigger calculation and check conditions for Indication or Notification to the Client based on the measurement value and descriptor settings.

The Application is responsible for storing the ES measurement value, triggers, trigger logic and client configuration information for each connection to a sensor instance. In addition the application also stores the user description (sensor name) for each characteristic instance. The measurement descriptor and valid range are stored in the GATT database.

During server initialization the current configuration is checked to determine if more than one trigger is present. If more than one trigger exists in the declaration of a characteristic – then a configuration descriptor (trigger logic) is added. If only a single trigger descriptor is present then the configuration descriptor would be automatically masked off..

Changes to measurement descriptor, trigger settings, configuration and user description are fully under the control of the application (meaning the application has to authorize any changes to these descriptors). Following changes to any of these descriptors – the application may trigger Indication messages (descriptor value change indication) to the impacted clients.

New measurements are compared to the corresponding trigger settings and configuration and if the trigger conditions are met, a new ES value is stored. If the notifications are enabled for a connection – then the new ES Value is notified to the client.

NOTE :- As the Measurement Descriptors and Valid Range for a given Sensor Characteristic are common to all links – it is essential that they are configured by the Application (in the ATT DB) prior to the establishment of any links to the server.

2.2.2 Modification of Descriptors from the Client:

Depending on the Server R/W Configuration, bonded clients should be able to modify following Descriptors: Client Characteristic Configuration (CCC)

- Trigger Settings
- Configuration (trigger logic)
- Client Characteristic Configuration (CCC)
- User Description

Requests for changes to these descriptors will be sent in the corresponding message ENVS_WR_XXX_REQ_IND to the APP_TASK. Changes to any of these are stored in the APP and will result in an updated triggering mechanism or user description.

3 API Messages

3.1.1 Initialization

During the initialization phase of the Environmental Sensing Server, the memory for this task must be allocated using the message GAPM_PROFILE_TASK_ADD_CMD provided by the GAPM interface. In addition to the security level, the following parameters should be provided:

Type	Parameters	Description
uint8_t	option	For future extension – Keep it 0
uint8_t	nb_chars	Number of Characteristic to appear in the profile
struct envs_char_instance	Instance[nb_chars]	An array of characteristic instances – format described in following table.

Table 16 – Parameter request for the GAPM_PROFILE_TASK_ADD_CMD

Type	Parameters	Description
uint8_t	es_char_id	characteristic type (see Table 1)
uint8_t	desc_en	A bitfield which defines the descriptors enabled for the Characteristic. Enabling the relevant bit enables the descriptor 0x01 - ENVS_DESC_EN_MEAS, 0x02 - ENVS_DESC_EN_TRIG_1 0x04 - ENVS_DESC_EN_TRIG_2 0x08 - ENVS_DESC_EN_TRIG_3 0x10 - ENVS_DESC_EN_CFG 0x20 - ENVS_DESC_EN_USR_DESC 0x40 - ENVS_DESC_EN_RANGE
uint8_t	desc_rw	This byte control the write configuration for the Characteristic Descriptors: 0x00 - ENVS_FEATURE_RO // ALL descriptors RO 0x01 - ENVS_FEATURE_WR_DESC // WR to User Description 0x02 - ENVS_FEATURE_WR_TRIG // WR to Trigger 0x03 - ENVS_FEATURE_WR_DESC_TRIG // WR to User Description and Trigger

Table 17 – Format of the envs_char_instance structure

Description: This API message shall be used to add one instance of the Environmental Sensing Service in the database.

Note: *The first element [0] of this configuration has to be the "ENVP_ES_DESCRIPTOR_VALUE_CHANGED_CHAR" Characteristic. This characteristic can be only single instance in the database and it can be disabled by selecting desc_en=0. Write enable for this Characteristic is not configurable so desc_rw is not interpreted.*

3.1.1.1 Example of Configuration

This sections show the configuration of a simple Environmental Sensing server – with both Humidity and Temperature sensors. Two instances of the temperature sensor are present, and all descriptors are enabled for each sensor.

```
/* Sample code for configuration of ENVS with 2 Temps Sensors + 1 Humidity Sensor + DVC
*/

#define ALL_DESCRIPTORS (ENVS_DECR_EN_MEAS|
                         ENVS_DECR_EN_TRIG_1|
                         ENVS_DECR_EN_TRIG_2|
                         ENVS_DECR_EN_TRIG_3|
                         ENVS_DECR_EN_CFG|
                         ENVS_DECR_EN_USR_DESC|
                         ENVS_DECR_EN_RANGE)


struct envs_db_cfg envs_db = {0, 4,
      {{ ENVP_ES_DESCRIPTOR_VALUE_CHANGED_CHAR, ALL_DESCRIPTORS, 0},
       { ENVP_ES_TEMPERATURE_CHAR,  ALL_DESCRIPTORS, ENVS_FEATURE_WR_DESC_TRIG},
       { ENVP_ES_TEMPERATURE_CHAR,  ALL_DESCRIPTORS,  ENVS_FEATURE_WR_DESC_TRIG},
       { ENVP_ES_HUMIDITY_CHAR,  ALL_DESCRIPTORS ,  ENVS_FEATURE_WR_DESC_TRIG}}};
```

This configuration code will result in the Characteristics and Descriptors shown in Figure 2

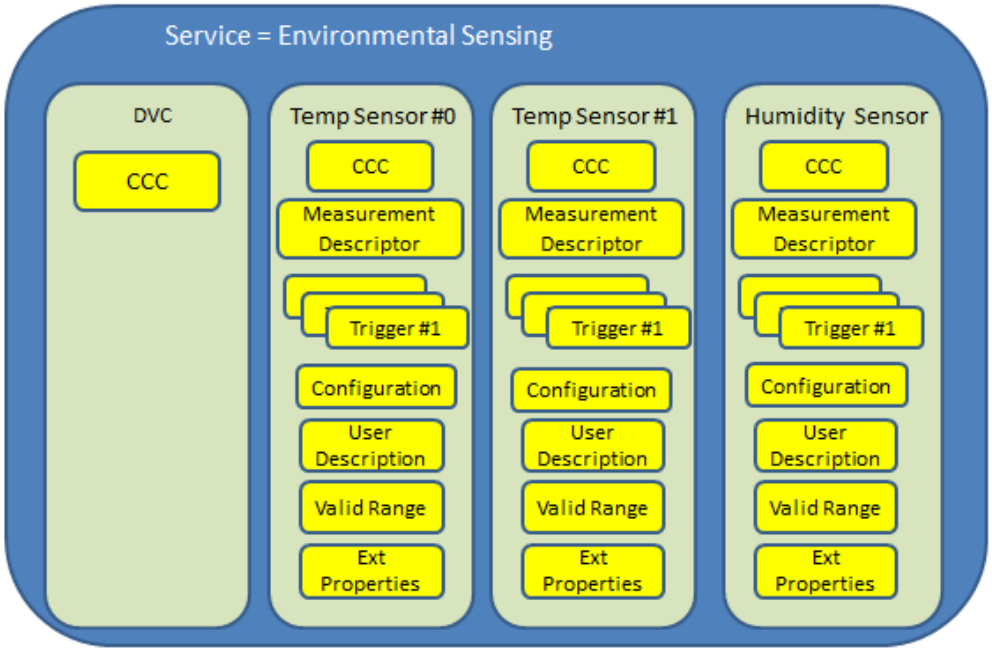

Figure 2 – Example configuration of 3 ES sensors

3.1.1.2 Setup of the Server Sequence

Some of the Characteristic descriptors carry nearly constant static value and should be configured at the start during initialization of the server.

The sequence of the initialization is as follow. Using the previous example configuration:

1. Create the service database:
   GAPM_PROFILE_TASK_ADD_CMD (struct envs_db_cfg)
2. Request to update the measurement descriptor:
   *ENVS_UPD_MEAS_CMD(ENVP_ES_TEMPERATURE_CHAR, 0/*inst 0 */,(struct envs_upd_meas_cmd))*

3. Request to change Valid Range descriptor:
   *ENVS_UPD_RANGE_CMD(ENVP_ES_TEMPERATURE_CHAR, 0/*inst 0 */,(struct envs_upd_range_cmd))*
4. Repeat steps 2 and 3 for the rest Characteristics:
   a. ENVP_ES_TEMPERATURE_CHAR, 1/*instance 1 */
   b. ENVP_ES_HUMIDITY_CHAR, 0/*instance 0 */

3.1.2 ENVS_UPD_MEAS_CMD

Source: TASK_APP

Destination: TASK_ENVS

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the ES characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the ES Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
struct envp_es_meas_desc	meas	The information to be stored in the measurement descriptor. Format is described in section 1.4, Table 3.

Response: ENVS_CMP_EVT

Description: This message is used to update the measurement descriptor for a given Sensor instance. The sensor is uniquely identified using the "es_char_id" and "es_char_inst". The measurement descriptor is not linked to any specific connection and cannot be modified on a per-connection basis. It is normally required if more than one instance of a specific sensor exist, and is used to all the client application distinguish between them.

NOTE :- This command should normally be used to update the measurement descriptor after Server initialization and before any connections are active. As it is not connection based – it should be addressed to TASK_ENVS using conidx = 0.

3.1.3 ENVS_UPD_RANGE_CMD

Source: TASK_APP

Destination: TASK_ENVS

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
union envp_range_char	range	Union of range values. The appropriate field of the union is selected based on the value of es_char_id. For the format of this union see section 1.8

Response: ENVS_CMP_EVT

Description :- The message is used to update the range of values supported by a given sensor instance. The range is common to all connections – and cannot be modified on a per-connection basis. Thus addressing this message to a specific connection will result in the range being modified for all connections.

NOTE :- This command should normally be used to update the measurement descriptor after Server initialization and before any connections are active. As it is not connection based – it should be addressed to TASK_ENVS using conidx = 0.

3.1.4 ENVS_RD_VALUE_REQ_IND

Source: TASK_ENVS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.

Response :- ENVS_RD_VALUE_CFM

Description :- This message is used to request the ES value from a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

3.1.5 ENVS_RD_VALUE_CFM

Source: TASK_APP

Destination: TASK_ENVS

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
union envp_val_char	value	Union containing the ES Value. The correct field of the union is selected based on the value of *es_char_id*

Response :- none

Description :- This message is used to provide a response to the ENVS_RD_VALUE_REQ_IND containing the current ES Value for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

A detailed description of the format of *union envp_val_char* can be found in section 1.9

3.1.6 ENVS_RD_CCC_REQ_IND

Source: TASK_ENVS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.

Response :- ENVS_RD_CCC_CFM

Description :- This message is used to request the CCC value from a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

3.1.7 ENVS_RD_CCC _CFM

Source: TASK_APP

Destination: TASK_ENVS

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
Uint16_t	ccc	Value of the CCC setting
uint8_t	Status	Indicates success/failure of the read operation

Response :- none

Description :- This message is used to provide a response to the ENVS_RD_CCC_REQ_IND containing the CCC flags.

3.1.8 ENVS_RD_TRIGGER_REQ_IND

Source: TASK_ENVS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
Uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint8_t	triggerIdx	The trigger instance if more than one trigger supported for this sensor instance. Valid values are 0,1,2.. If only a single trigger is supported the value is 0.

Response :- ENVS_RD_TRIGGER_CFM

Description :- This message is used to request a trigger setting for a given trigger (identified by *triggerIdx*), from a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

3.1.9 ENVS_RD_TRIGGER_CFM

Source: TASK_APP

Destination: TASK_ENVS

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint8_t	triggerIdx	The trigger instance if more than one trigger supported for this sensor instance. Valid values are 0,1,2
uint8_t	status	Indicates success/failure of the read operation
uint8_t	condition	Trigger condition value (see Table 5)
uint32_t	time_based_trigger	Value of a time based trigger – applicable only if condition = 1 or 2, otherwise set to Zero.
union envp_trigger_val_char	meas_value_trigger	A union of measurement value triggers, with the relevant field being selected based on the value of *es_char_id.* This field is only relevant if the condition has a value between 4 and 9 (inclusive), otherwise this should be set to zero.

Response :- none

Description :- This message is used to provide a response to the ENVS_RD_TRIGGER_REQ_IND containing the trigger setting, for a given trigger instance (identified by triggerIdx) for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

Detailed description of the parameters used to define the trigger can be found in section 1.5.

3.1.10 ENVS_RD_CONFIG_REQ_IND

Source: TASK_ENVS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.

Response :- ENVS_RD_CONFIG_CFM

Description :- This message is used to request the configuration value (trigger logic) for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

3.1.11 ENVS_RD_CONFIG _CFM

Source: TASK_APP

Destination: TASK_ENVS

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
Uint8_t	trigger_logic	Value of the configuration setting (0 – And, 1 – Or)
uint8_t	Status	Indicates success/failure of the read operation

Response :- none

Description :- This message is used to provide a response to the ENVS_RD_CONFIG_REQ_IND containing the configuration (trigger logic) for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

3.1.12 ENVS_RD_USER_DESCRIPTION_REQ_IND

Source: TASK_ENVS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.

Response :- ENVS_RD_USER_DESCRIPTION_CFM

Description :- This message is used to request the user description (utf 8 name) for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

3.1.13 ENVS_RD_USER_DESCRIPTION_CFM

Source: TASK_APP

Destination: TASK_ENVS

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
struct utf_8_name	utf_name	A UFT_8 string which becomes the name of a given Sensor instance
uint8_t	Status	Indicates success/failure of the read operation

Response :- none

Description :- This message is used to provide a response to the ENVS_RD_USER_DESRIPTION_REQ_IND containing the user description (utf 8 name) for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

Format of the utf_8_name structure is :

```
struct utf_8_name
{
    uint8_t length;
    uint8_t name[__ARRAY_EMPTY];
};
```

3.1.14 ENVS_WR_TRIGGER_REQ_IND

Source: TASK_ENVS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint8_t	triggerIdx	The trigger instance if more than one trigger supported for this sensor instance. Valid values are 0,1,2
uint8_t	condition	Trigger condition value (see section X.Y)
uint32_t	time_based_trigger	Value of a time based trigger – applicable only if condition = 1 or 2, otherwise set to Zero.
union envp_trigger_val_char	meas_value_trigger	A union of measurement value triggers, with the relevant field being selected based on the value of *es_char_id.* This field is only relevant if the condition has a value between 4 and 9 (inclusive), otherwise this should be set to zero.

Response: ENVS_WR_TRIGGER_CFM

Description :- This message is used to request a change in the trigger settings to the application, for a given trigger instance (identified by triggerIdx) for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

Detailed description of the parameters used to define the trigger can be found in section 1.5.

3.1.15 ENVS_WR_TRIGGER_CFM

Source: TASK_ENVS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic which was written to.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint8_t	status	Indicates success/failure of the write to the application

Response: none

Description :- This message is used to provide a response to a previous ENVS_WR_TRIGGER_REQ_IND. a change in the trigger settings to the application, for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*.

3.1.16 ENVS_WR_CONFIG_REQ_IND

Source: TASK_ENVS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint8_t	trigger_logic	Value of the configuration setting (0 – And, 1 – Or)

Response: ENVS_WR_CONFIG_CFM

Description :- This message is used to request a change in the configuration (trigger logic) to the application, for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

If more than one trigger is active for a Characteristic, then the trigger logic is used to determine how to combine the effects of a trigger. If multiple triggers are not present this message should be rejected.

Trigger logic	Value	Description
ENVS_BOOL_AND	0x00	Perform a Boolean AND of all available active triggers.
ENVS_BOOL_OR	0x01	Perform a Boolean OR of all available active triggers.

3.1.17 ENVS_WR_CONFIG_CFM

Source: TASK_APP

Destination: TASK_ENVS

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic which was written to.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more

		than one instance of the same type exist.. Otherwise this is left at '0'.
uint8_t	status	Indicates success/failure of the write to the application

Response: none

Description :- This message is used to provide a response to a previous ENVS_WR_CONFIG_REQ_IND to request a change in the configuration (trigger logic), for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*.

3.1.18 ENVS_WR_USER_DESCRIPTION_REQ_IND

Source: TASK_ENVS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
struct utf_8_name	utf_name	The string used to describe the device. The format of this structure was described in a preceding section.

Response: ENVS_WR_USER_DESCRIPTION_CFM

Description: This message is used to request the application to update the User Description to the name given in the *utf_name* parameter of the message.

NOTE : "In order for writes to the Characteristic User Description descriptor to be enabled, the Bluetooth Core Specification [1] requires that the associated characteristic has the Extended Properties property, the Characteristic Extended Properties descriptor is present, and the Writable Auxiliaries bit of Characteristic Extended Properties descriptor is set to 1."

3.1.19 ENVS_WR_USER_DESCRIPTION_CFM

Source: TASK_APP

Destination: TASK_ENVS

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic which was written to.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint8_t	status	Indicates success/failure of the write to the application

Response: none

Description :- This message is used to provide a response to a previous ENVS_WR_USER_DESCRIPTION_REQ_IND to request a change in the name for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

3.1.20 ENVS_WR_CCC _REQ_IND

Source: TASK_ENVS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint16_t	ccc	The new values requested for the CCC . (valid values 0,1,2)

Response: ENVS_WR_CCC_CFM

Description: This message is used to request the application to update the Client Characteristic Configuration for a given sensor instance (identified by *es_char_id* and *es_char_inst*). Only a subset of values are legal, depending on the characteristic type identified in *es_char_id*.

es_char_id	es_char_id value	Valid CCC values
DESCRIPTOR_VALUE_CHANGED	0x00	0x00 – No Indications, 0x02 Indicate
Any other ES Char	0x01-0x14	0x00 – No Notifications – 0x01 Notifications

3.1.21 ENVS_WR_CCC_CFM

Source: TASK_APP

Destination: TASK_ENVS

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic which was written to.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint8_t	status	Indicates success/failure of the write to the application

Response: none

Description :- This message is used to provide a response to a previous ENVS_WR_CCC_REQ_IND to request a change in the client characteristic configuration (control of notifications/indications), for a particular sensor instance, identified by *es_char_id* and *es_char_inst*..

3.1.22 ENVS_NOTIFY_CMD

Source: TASK_APP

Destination: TASK_ENVS

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic which the client is being notified about.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
union envp_val_char	value	Union containing the ES Value. The correct field of the union is selected based on the value of *es_char_id*

Response: ENVS_CMP_EVT

Description: This message is used to update the client of a change in an ES value for the sensor instance identified by *es_char_id* and *es_char_inst*. The format of *union envp_val_char* is described in section 1.9.

***Note 1** :- Unlike other profiles where the profile sends notifications to all connected clients (with the CCC enabled). The distribution of update ES Values to multiple clients is performed by the Application not the profile. This is due to the trigger settings, trigger logic and CCC (which can be different for each client) being located with the Application (not the profile). Thus only the application can determine if a new measurement meets the conditions imposed by the trigger settings and logic for each connection, and determine if the CCC is enabled for notifications.*

***Note 2**:- This command is not applicable to the ENVP_ES_DESCRIPTOR_VALUE_CHANGED_CHAR (es_char_id = 0).*

3.1.23 ENVS_INDICATE_CMD

Source: TASK_APP

Destination: TASK_ENVS

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic whose descriptors have changed and should be re-read by the client.
uint16_t	Ind_flag	This field indicates what has changed and the source of the changes. More information on the format of this field can be found in section 1.7

Response: ENVS_CMP_EVT

Description: This message is used to update the client that either the trigger settings, trigger configuration (trigger logic), measurement descriptor or user description has changed. The *ind_flag* field indicates what has changed and the source of the change.

If the trigger settings or configurations changes this command should only be sent to the client on the effecting link (as trigger setting are link specific). However, if the measurement (descriptor) or user description are changed by the server this should be indicated to all clients. Similarly, if the user description is changed by a client, the modification must be indicated to all clients – except the client which made the modification.

NOTE :- In a similar manner to Notifictions, this command only sends an indication to a single client. If multiple clients are to be sent an Indication, then it is the responsibility of the Application to call this command for each client to receive the notification. The reason for this behavior is the connection based CCC used to control the indications (the CCC of the Descriptor Value Changed Characteristic) is under the control of the Application.

NOTE :- For bonded clients which do not have an active connection when a descriptor value changes, the server should send a Indication when they re-connect.

3.1.24 ENVS_CMP_EVT

Source: TASK_ENVS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	Operation	Operation Code:
uint8_t	Status	Status

Description: The API message is used by the ENVS task to inform the sender of a command that the procedure is complete in the server profile and returns the status field indicating the outcome of the procedure.

The valid operation codes are the following :

Operation Code	VALUE
ENVC_RESERVED_OP_CODE	0x00
ENVC_UPD_MEAS_DESCRIPTOR_OP_CODE	0x01
ENVC_UPD_USER_DESCRIPTION_OP_CODE	0x02
ENVC_UPD_RANGE_OP_CODE	0x03
ENVS_NOTIFY_OP_CODE	0x04
ENVS_INDICATE_OP_CODE	0x05

4 ENVC Collector Role API

Within the ENVC task, three main states are defined: FREE, IDLE, DISCOVERYING and BUSY.

The TASK_ENVC is multi-instantiated with one instance created for each connection for which the profile will be enabled and each of these instances will have a different task ID..

4.1 API Messages

4.1.1 Initialization

During the initialization phase of the Environmental Sensing Collector, the memory for this task must be allocated using the message GAPM_PROFILE_TASK_ADD_CMD provided by the GAPM interface.

4.1.2 ENVC_ENABLE_REQ

Source: TASK_APP

Destination: TASK_ENVC

Parameters:

Type	Parameters	Description
uint8_t	con_type	Connection type (PRF_CON_DISCOVERY (0x00) or PRF_CON_NORMAL (0x01)
struct envc_es_content	es	Provides service details and information on all previously discovered characteristics and descriptors for peer server.

Response :- ENVC_RD_VALUE_RSP

Description: This API message is used for enabling the Collector role of theENVC. This message contains the connection type and optionally any previously saved discovered ENVC details on peer. The connection type may be PRF_CON_DISCOVERY (0x00) for discovery/initial configuration or PRF_CON_NORMAL (0x01) for a normal connection with a bonded device..

For a discovery connection, discovery of the peer ENVS is started and an ENVC_ES_INFO_IND is sent for each discovered characteristic. The ENVC_ES_INFO_IND contains all the information which needs to be saved for this characteristic, to be successfully cached for subsequent connections. When all the characteristics of the peer ENVS server have been discovered an ENVC_ENABLE_RSP event is sent.

Format of *envc_es_content*

Type	Parameters	Description
struct prf_svc	svc	The start and end handle of the service
uint8_t	nb_chars_discovered	Total number of characteristic discovered, this should be equivalent to the number of ENVC_ES_INFO_IND events received.
struct envc_es_char_info	chars[ENVC_ES_MAX_CHARS_ALLOWED]	Array describing each of the discovered characteristic. Each element should map directly to the information received in a ENVC_ES_INFO_IND.

Format of *envc_es_char_info is described in section 1.10 and table 14.*

Figure 3 shows the API messages exchanged during discovery, triggered by an ENVC_ENABLE_REQ with con_type set to PRF_CON_DISCOVERY.

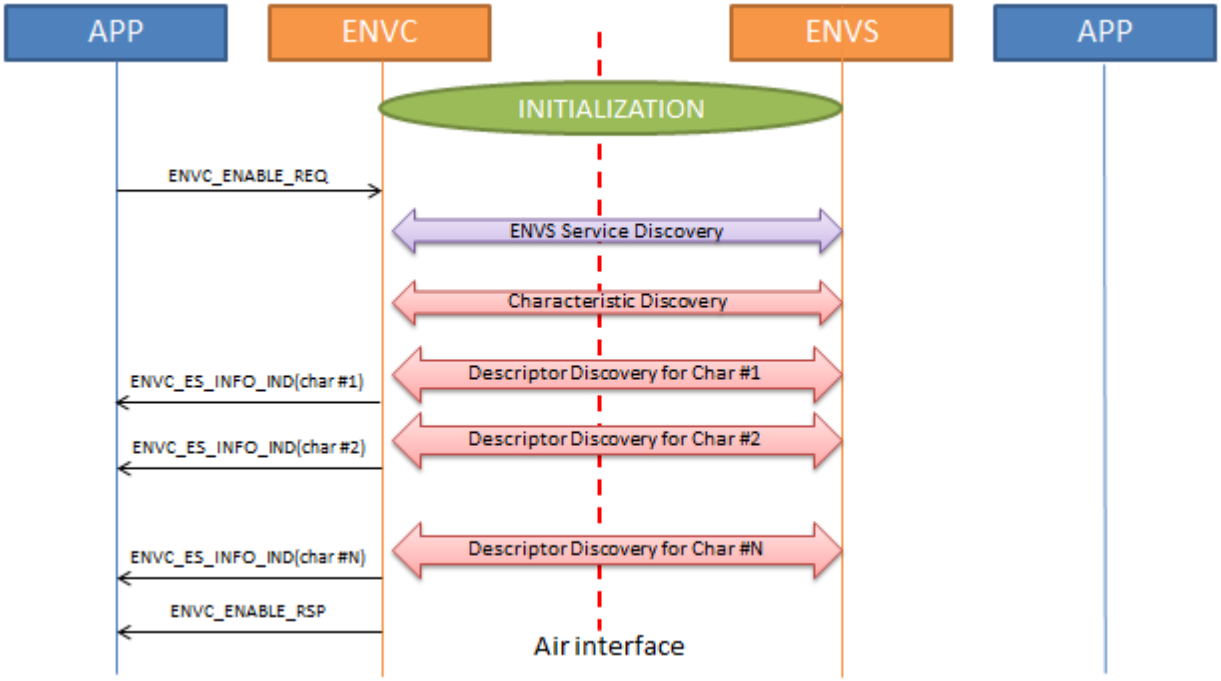

Figure 3 :- Initial Discovery

4.1.3 ENVC_ENABLE_RSP

Source: TASK_APP

Destination: TASK_ENVC

Parameters:

Type	Parameters	Description
uint8_t	Status	Indicates Success or Failure of the operation.
struct prf_svc	Svc	The start and end handle of the service – if the original connection type was PRF_CON_DISCOVERY.
Uint8_t	nb_chars_discovered	Total number of characteristic discovered, this should be equivalent to the number of ENVC_ES_INFO_IND events received. Only applicable if the original connection type was PRF_CON_DISCOVERY.

Response :- none

Description :- if the original connection type was PRF_CON_DISCOVERY this response provide the information used to create the *envc_es_content* in combination with the information obtained in the ENVC_ES_INFO_INDs.

4.1.4 ENVC_ES_INFO_IND

Source: TASK_APP

Destination: TASK_ENVC

Parameters:

See section 1.10 – table 14.

Response :- none

Description :- This event is used to present all the information on a characteristic to the application. The handles for the characteristic value and all the characteristic descriptors are provided.

4.1.5 ENVC_RD_VALUE_REQ

Source: TASK_APP

Destination: TASK_ENVC

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.

Response :- ENVC_RD_VALUE_RSP

Description :- This message is used to request the ES value from a particular sensor instance, identified by *es_char_id* and *es_char_inst* from the server.

4.1.6 ENVC_RD_VALUE_RSP

Source: TASK_ENVC

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
union envp_val_char	value	Union containing the ES Value. The correct field of the union is selected based on the value of *es_char_id*

Response :- none

Description :- This message is used to provide a response to the ENVS_RD_VALUE_REQ containing the current ES Value for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

A detailed description of the format of *union envp_val_char* can be found in section 1.9

4.1.7 ENVC_RD_NTF_CFG_REQ

Source: TASK_APP

Destination: TASK_ENVC

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic we need to retrieve the CCC for.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.

Response :- ENVC_RD_NTF_CFG_RSP

Description :- This message is used to request the CCC value from a particular sensor instance, identified by *es_char_id* and *es_char_inst.* from the server.

4.1.8 ENVC_RD_NTF_CFG _RSP

Source: TASK_ENVC

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint16_t	ccc	Value of the CCC setting
uint8_t	Status	Indicates success/failure of the read operation

Response :- none

Description :- This message is used to provide a response to the ENVS_RD_NTF_CFG_REQ containing the CCC flags.

4.1.9 ENVC_RD_TRIGGER_REQ

Source: TASK_APP

Destination: TASK_ENVC

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic.
Uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint8_t	triggerIdx	The trigger instance if more than one trigger supported for this sensor instance. Valid values are 0,1,2.. If only a single trigger is supported the value is 0.

Response :- ENVC_RD_TRIGGER_RSP

Description :- This message is used to request a trigger setting for a given trigger (identified by *triggerIdx*), from a particular sensor instance, identified by *es_char_id* and *es_char_inst.* from the server.

4.1.10 ENVC_RD_TRIGGER_RSP

Source: TASK_ENVC

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint8_t	triggerIdx	The trigger instance if more than one trigger supported for this sensor instance. Valid values are 0,1,2
uint8_t	status	Indicates success/failure of the read operation
uint8_t	condition	Trigger condition value (see section X.Y)
uint32_t	time_based_trigger	Value of a time based trigger – applicable only if condition = 1 or 2,

		otherwise set to Zero.
union envp_trigger_val_char	meas_value_trigger	A union of measurement value triggers, with the relevant field being selected based on the value of *es_char_id.* This field is only relevant if the condition has a value between 4 and 9 (inclusive), otherwise this should be set to zero.

Response :- none

Description :- This message is used to provide a response to the ENVS_RD_TRIGGER_REQ containing the trigger setting, for a given trigger instance (identified by triggerIdx) for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*in the server

Detailed description of the parameters used to define the trigger can be found in section 1.5.

4.1.11 ENVC_RD_CONFIG_REQ

Source: TASK_APP

Destination: TASK_ENVC

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.

Response :- ENVC_RD_CONFIG_RSP

Description :- This message is used to request the configuration value (trigger logic) for a particular sensor instance, identified by *es_char_id* and *es_char_inst* .from the server.

4.1.12 ENVC_RD_CONFIG_RSP

Source: TASK_ENVC

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic requested.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
Uint8_t	trigger_logic	Value of the configuration setting (0 – And, 1 – Or)
uint8_t	Status	Indicates success/failure of the read operation

Response :- none

Description :- This message is used to provide a response to the ENVS_RD_CONFIG_REQ containing the configuration (trigger logic) for a particular sensor instance, identified by *es_char_id* and *es_char_inst,* from the server.

4.1.13 ENVC_RD_USER_DESCRIPTION_REQ

Source: TASK_APP

Destination: TASK_ENVC

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.

Response :- ENVS_RD_USER_DESCRIPTION_CFM

Description :- This message is used to request the user description (utf 8 name) for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

4.1.14 ENVC_RD_USER_DESCRIPTION_RSP

Source: TASK_ENVC

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
struct utf_8_name	utf_name	A UFT_8 string which becomes the name of a given Sensor instance
uint8_t	Status	Indicates success/failure of the read operation

Response :- none

Description :- This message is used to provide a response to the ENVS_RD_USER_DESRIPTION_REQ containing the user description (utf 8 name) for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

4.1.15 ENVC_RD_RANGE _REQ

Source: TASK_APP

Destination: TASK_ENVC

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.

Response :- ENVC_RD_RANGE_RSP

Description :- This message is used to request the valid range for a particular sensor instance, identified by *es_char_id* and *es_char_inst* from the server.

4.1.16 ENVC_RD_RANGE_RSP

Source: TASK_ENVC

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	status	Indicates success/failure of the request.
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.

uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
union envp_range_char	Range	Union of range values. The appropriate field of the union is selected based on the value of es_char_id

Response: none

Description :- The message returns the result to a ENVS_RD_USER_DESRIPTION_REQ which contains the range of values supported by a given sensor instance.

4.1.17 ENVC_RD_EXT_PROP_REQ

Source: TASK_APP

Destination: TASK_ENVC

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.

Response :- ENVC_RD_EXT_PROP_RSP

Description :- This message is used to request the external properties for a particular sensor instance, identified by *es_char_id* and *es_char_inst* from the server.

4.1.18 ENVC_RD_EXT_PROP_RSP

Source: TASK_ENVC

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	status	Indicates success/failure of the request.
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint16	ext_properties	The extended properties, only valid value are 0 & 2. A value 2 indicates that the User Description is writable.

Response: none

Description :- The message returns the result to a ENVC_RD_EXT_PROP_REQ which contains the external properties supported by a given sensor instance. The extended properties is only required to determine if the User Description descriptor is writable.

4.1.19 ENVC_RD_MEAS_DESCRIPTOR_REQ

Source: TASK_APP

Destination: TASK_ENVC

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.

Response :- ENVC_RD_MEAS_DESCRIPTOR_RSP

Description :- This message is used to request the measurement descriptor for a particular sensor instance, identified by *es_char_id* and *es_char_inst* from the server.

4.1.20 ENVC_RD_MEAS_DESCRIPTOR_RSP

Source: TASK_ENVC

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	status	Indicates success/failure of the request.
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
struct envp_es_meas_desc	meas	The information stored in the measurement descriptor. Format is described in section 1.4, Table 3.

Response: none

Description :- The message returns the result to a ENVC_RD_MEAS_DESCRIPTOR_REQ which contains the contents of the measurement descriptor for a given sensor instance. Further information on the format of this information and the structure of *struct envp_es_meas_desc* can be found in section 1.4.

4.1.21 ENVC_WR_TRIGGER_REQ

Source: TASK_APP

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint8_t	triggerIdx	The trigger instance if more than one trigger supported for this sensor instance. Valid values are 0,1,2
uint8_t	condition	Trigger condition value (see section X.Y)
uint32_t	time_based_trigger	Value of a time based trigger – applicable only if condition = 1 or 2, otherwise set to Zero.
union envp_trigger_val_char	meas_value_trigger	A union of measurement value triggers, with the relevant field being selected based on the value of *es_char_id.* This field is only relevant if the condition has a value between 4 and 9 (inclusive), otherwise this should be set to zero.

Response: ENVC_CMP_EVT

Description :- This message is used to request a change in the trigger settings to the server, for a given trigger instance (identified by triggerIdx) for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

Detailed description of the parameters used to define the trigger can be found in section 1.5.

4.1.22 ENVC_WR_CONFIG_REQ

Source: TASK_APP

Destination: TASK_ENVC

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint8_t	trigger_logic	Value of the configuration setting (0 – And, 1 – Or)

Response: ENVC_CMP_EVT

Description :- This message is used to request a change in the configuration (trigger logic) in the server, for a particular sensor instance, identified by *es_char_id* and *es_char_inst.*

If more than one trigger is active for a Characteristic, then the trigger logic is used to determine how to combine the effects of a trigger. If multiple triggers are not present this message should be rejected.

Trigger logic	Value	Description
ENVS_BOOL_AND	0x00	Perform a Boolean AND of all available active triggers.
ENVS_BOOL_OR	0x01	Perform a Boolean OR of all available active triggers.

4.1.23 ENVC_WR_USER_DESCRIPTION_REQ

Source: TASK_APP

Destination: TASK_ENVC

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
struct utf_8_name	utf_name	The string used to describe the device. The format of this structure was described in a preceding section.

Response: ENVS_CMP_EVT

Description: This message is used to request the server to update the User Description to the name given in the *utf_name* parameter of the message.

NOTE : "In order for writes to the Characteristic User Description descriptor to be enabled, the Bluetooth Core Specification [1] requires that the associated characteristic has the Extended Properties property, the Characteristic Extended Properties descriptor is present, and the Writable Auxiliaries bit of Characteristic Extended Properties descriptor is set to 1."

4.1.24 ENVC_WR_NTF_CFG_REQ

Source: TASK_APP

Destination: TASK_ENVC

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic to be updated.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
uint16_t	value	This field hold the value to be written to the CCC descriptor . Only values 0,1,2 are allowed. If the es_char_id identifies a Descriptor Value Change characteristic – then this value is used to turn on/off indications. If any other characteristic is chosen then this value is used to turn on/off ES Value notifications from the server..

Response: ENVC_CMP_EVT

Description: This message is used to control the delivery of Notifications and Indications from a sensor instance on the server. If the es_char_id selected is the ENVP_ES_DESCRIPTOR_VALUE_CHANGED_CHAR, then this request is used to turn on/off Descriptor Value Changed (DVC) indications from the server (0x0 – off, 0x2- on).

If es_char_id identifies any other valid sensor – then this command is used to turn on/off the notifications of ES Value changes from the server (0x0 – off, 0x01 – on).

4.1.25 ENVC_VALUE_IND

Source: TASK_ENVC

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic which the client is being notified about.
uint8_t	es_char_inst	Identifies the instance of the Measurement Characteristic IF more than one instance of the same type exist.. Otherwise this is left at '0'.
union envp_val_char	value	Union containing the ES Value. The correct field of the union is selected based on the value of *es_char_id*

Response: ENVS_CMP_EVT

Description: This message is used to update the client of a change in an ES value for the sensor instance identified by *es_char_id* and *es_char_inst*. The format of *union envp_val_char* is described in section 1.9.

Note 1 *:- es_char_id = ENVP_ES_DESCRIPTOR_VALUE_CHANGED_CHAR (es_char_id = 0)* is illegal for this message

4.1.26 ENVC_DVC_IND

Source: TASK_ENVC

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	es_char_id	Identifier for the measurement characteristic whose descriptors has changed and should be re-read by the client.
uint16_t	flags	This field indicates what has changed and the source of the changes. More information on the format of this field can be found in section 1.7

Response: ENVC_CMP_EVT

Description: This message is used to update the client that either the trigger settings, trigger configuration (trigger logic), measurement descriptor or user description has changed. The *flags* field indicates what has changed and the source of the change.

The format of the flags field is described in Section 1.7

5 Message Sequence Charts (MSCs)

This part describes the different procedure that can be used within the Environmental Sensing profile. In these MSCs, it is supposed that two RW stacks (one with the server role of the profile and one with the client role) are connected together.

The example scenario is 3 ES Characteristics, consisting of two Thermometers (one air thermometer, one ground thermometer) and a Humidity sensor.

5.1 Setting up the Server

After initialization and prior to the establishment of any ACL links the Application should write to the ATT_DB to configure the different sensor information. For each sensor both the measurement descriptor and the range descriptor have to be configured for each of the ES Characteristics available. As can be seen in Figure 4, the First Thermometer is configured by writing to its measurement descriptor (using ENVS_UPD_MEAS_CMD), in this case the Application field of the measurement descriptor will distinguish it as an Air Thermometer (for simplicity the remaining fields of the measurement descriptor information are not shown). The valid range for this thermometer is then configured (using ENVS_UPD_RANGE_CMD) to have a range from -20c to +100c.

The second thermometer is then configured by updating the measurement descriptor to indicate a ground thermometer. Similarly the valid range is configure to support a range of -40c to +50c.

Finally, the descriptors for the Humidity characteristic are written. Indicating a measurement period of 10mins, and a valid range of 0 to 80%.

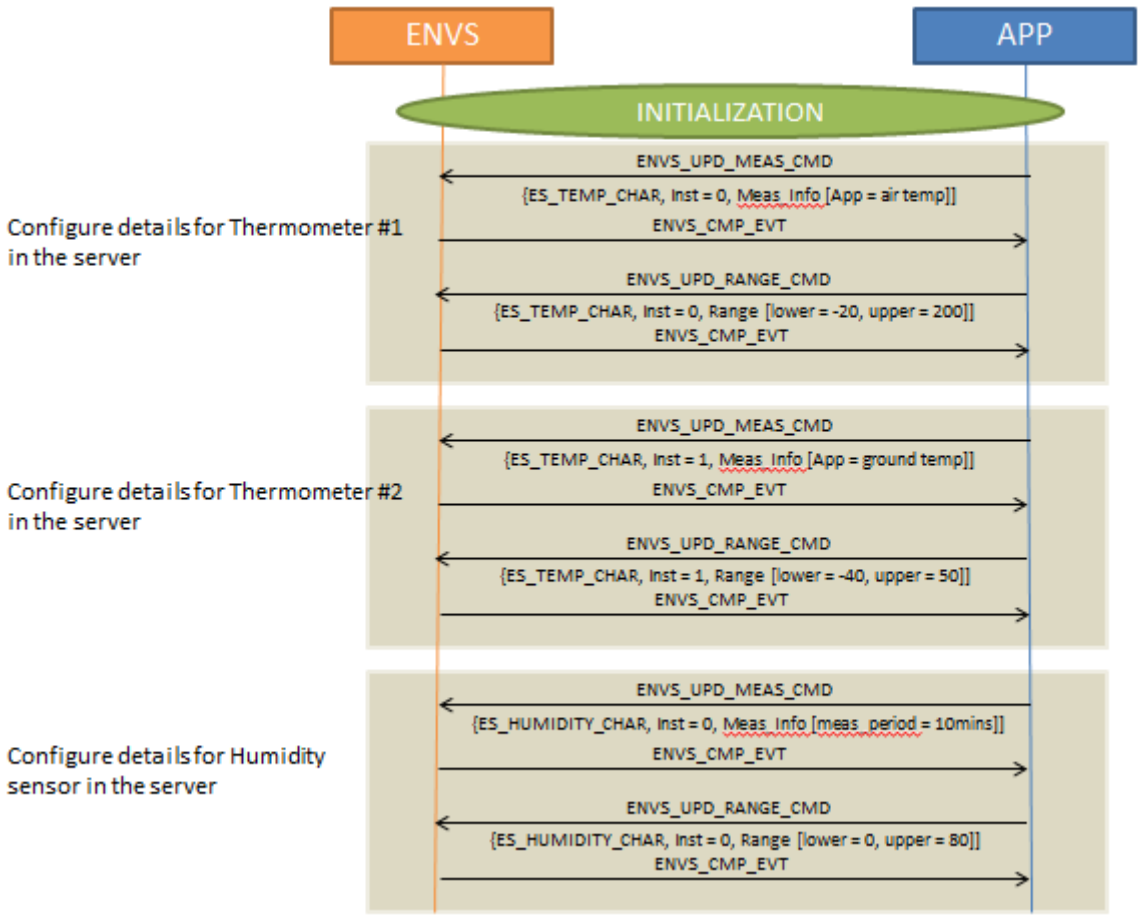

Figure 4 – Setting up the Server for 2 thermometers and a humidity sensor.

5.2 Discovery by the Client

Figure 5 shows the sequence of messages exchanged on the API during discovery by the client. The client application sends the ENVC_ENABLE_REQ and indicates discovery should be performed (con_type = discovery). This results in the ENVC performing service discovery, followed by characteristic discovery. For each discovered characteristic – a full descriptor discovery is performed. When each descriptor discovery is complete – the ENVC_ES_INFO_IND message is sent to the application containing the handles of characteristic value, and all associated descriptors. This is formation is not used directly by the application but can be cached and used in subsequent connections to bypass discovery.

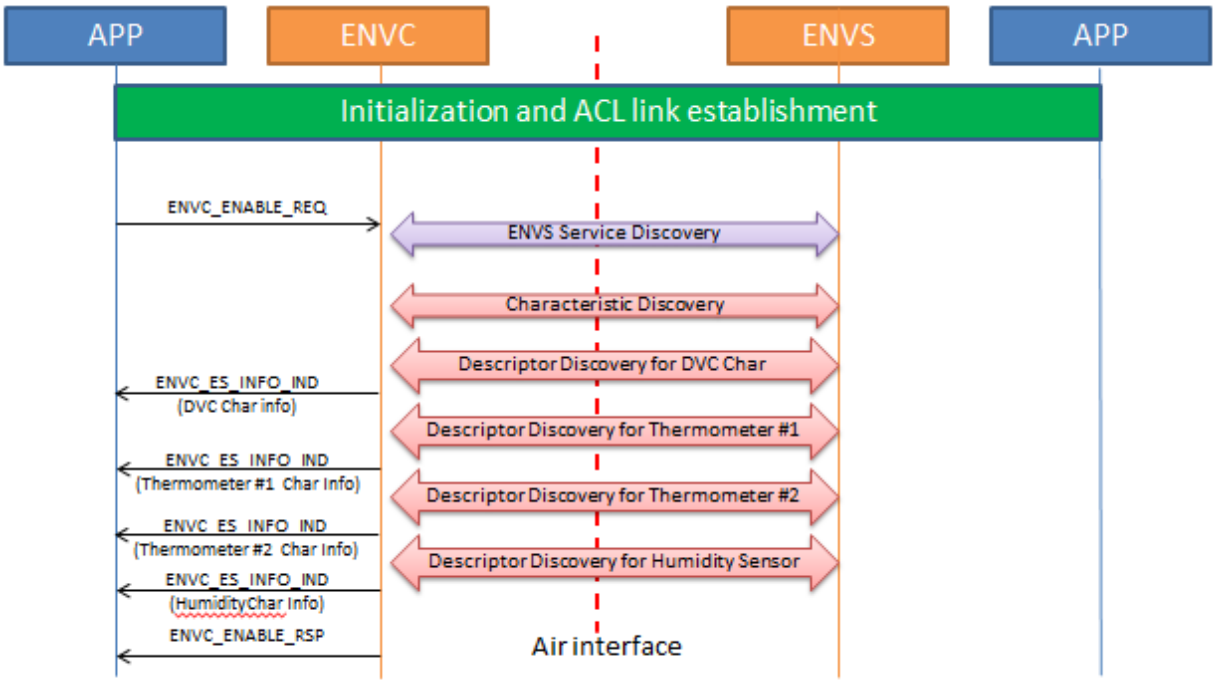

Figure 5 – Discovery of 2 Thermometers and a Humidity sensor by the client.

5.3 Client Configuration of the Server

In this example two thermometers are present in the server. To distinguish between them the client has to read the Measurement descriptor of each. Figure 6 shows the client reading the measurement descriptor of each of the 3 characteristic present in the server. Subsequently as, in this case, the client is not interested in ground temperature; it proceeds to configure the Client Characteristic Configuration (CCC) for the first thermometer and the humidity sensor, to ensure it is notified of air temperature and humidity changes.

The reading of the measurement descriptor is performed transparently to the application on the server. However, as the CCC is maintained by the application (and closely coupled to trigger information) the request to modify it is provided to the application.

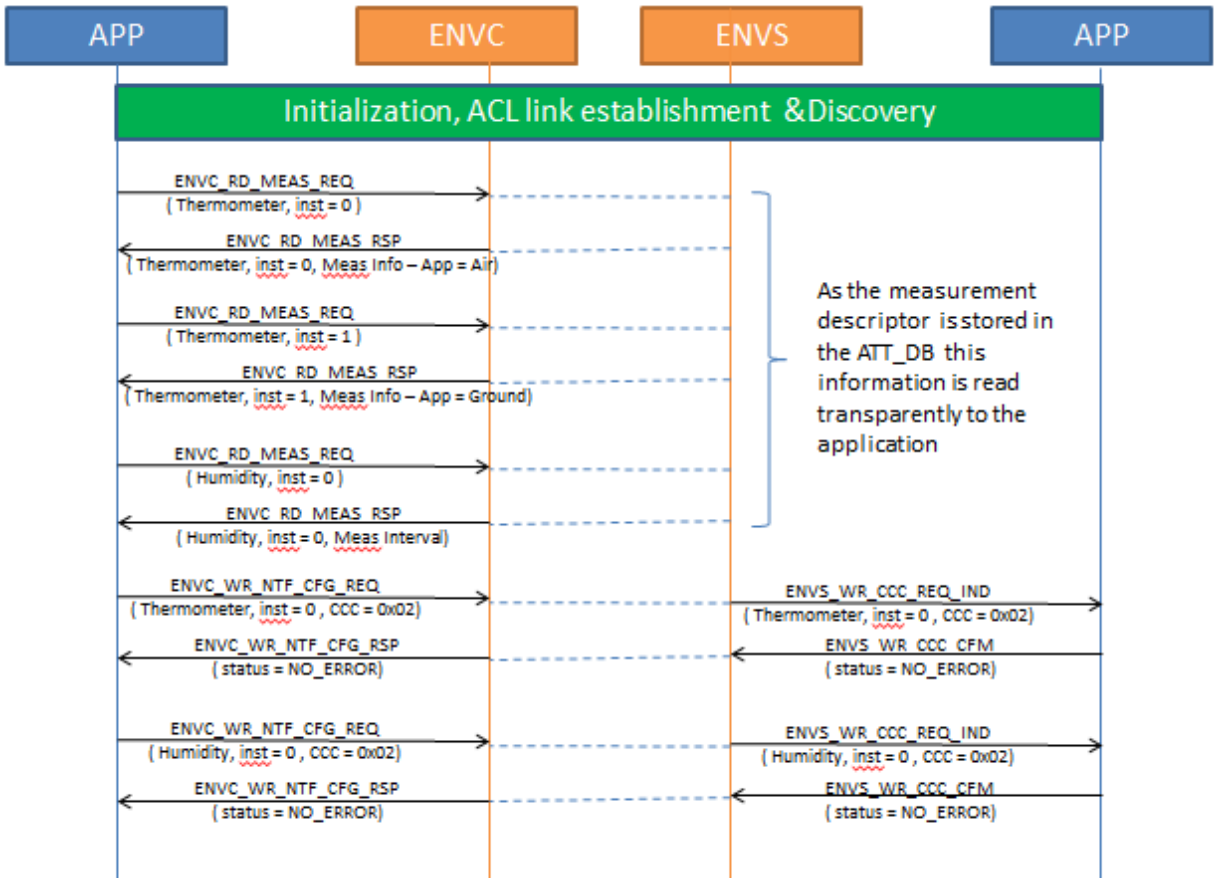

Figure 6 – Client Configuration of the Server.

5.4 Notification of Updated Values from the Server

Once the CCCs for each sensor have been configured for notifications, the client is notified of new values which satisfy the trigger conditions on the server (if triggers are present). Figure 7 shows the server application sending two notify commands (one for Thermometer #1 and one for humidity sensor). These result in a ENVC_VALUE_IND being sent to the Application on the client side.

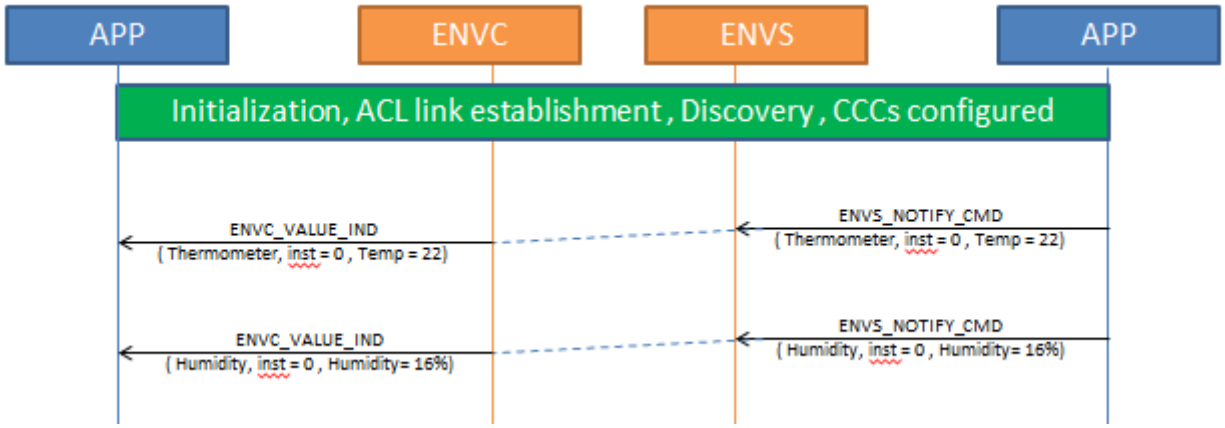

Figure 7 – Notification of updated values from Server.