RW BLE Cycling Power Profile Interface Specification

Interface Specification

RW-BLE-CPP-IS

Version 9.0

2017-03-09

RivieraWaves
http://www.rivierawaves.com/

Revision History

Version	Date	Revision Description	Author
0.1	2014-01-16	Initial draft	CM
1.0	2014-04-03	Initial release	CM
1.1	2014-04-14	Few relevant changes	KY
1.2	2014-05-13	Feature details	KY
7.0	2014-10-13	Updated for BLE 4.1	CM
8.0	2015-07-29	Updated for BLE 4.2	CM
9.0	2017-03-09	Updated for BLE 5	LT

Table of Contents

1 Overview

1.1 Document Overview

This document describes the non-standard interface of the RW Bluetooth Low Energy (BLE) Cycling Power Profile (CPP) implementation. In this document, the interface messages will be referred to as API messages for the profile.

Description will include the rationale behind CPP's design/implementation. This would provide better understanding to the user and/or developer for profile usage from upper layer or the final application.

1.2 BLE Cycling Power Profile Overview

The CPP enables a Cycling Power Collector to connect, interact and exchange information with a Cycling Power Sensor for use in sports and fitness applications.

The CPP has been implemented as an LE (GATT-based) profile. Within this profile, two roles can be supported: Sensor (CPPS) and Collector (CPPC). The Collector shall support the GAP's central role while the Sensor shall support the GAP's peripheral role. The profile first requires an LE connection to be established between the two devices before to realize the Cycling Power functionalities.

The documents edited by the Bluetooth SIG present different use cases for this profile, their GATT, GAP and SM, mandatory and optional requirements. The Cycling Power Profile specifications have been adopted by the Bluetooth SIG on April 30th 2013 ([1] and [3]). Their related test specifications have been released on December 3rd ([2] and [4]).

The profile is implemented in the RW-BLE software stack in two sub-blocks, one for each role. Each sub-block has a plurality of APIs decided after the study of the profile specifications and test specifications. The design is considered to be minimalistic and conceived for future complex application, which would combine the profile functionality with the device's connectivity and security procedures.

The structure of the CP service is defined as shown in the table below:

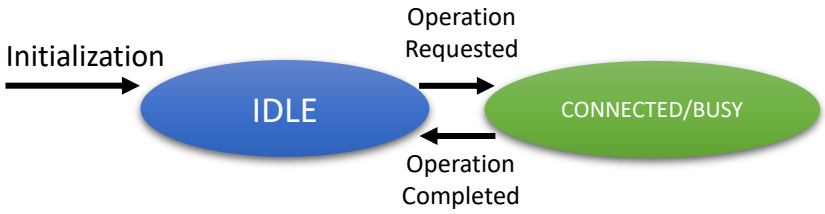
Characteristic Name	Requirements	Properties	Security	Descriptors
CP Measurement	Mandatory	Notify	None	Client / Server Characteristic Configuration
CP Feature	Mandatory	Read	None	None
Sensor Location	Mandatory	Read	None	None
CP Vector	Optional	Notify	None	Client Characteristic Configuration
CP Control Point	Optional	Write/Indicate	None	Client Characteristic Configuration

2 CPP Sensor Role API

2.1 Environment

This role should be activated in every application that a Cycling Power Sensor is required; the provided API is capable of sending notifications such as several power or vector measurements to the collector. This FW will behave as configured by the Collector in the CPP characteristic. Please, refer to "cpps_task.h" for implementation of this API.

Within the CPPS task, two states are defined: **IDLE and CONNECTED/BUSY.**



2.2 API Messages

2.2.1 Initialization/Database creation

During the initialization phase of the Cycling Power Sensor, the memory for this task must be allocated using the message GAPM_PROFILE_TASK_ADD_CMD provided by the GAPM interface. Apart from the security level, the following parameters should be filled:

Parameters:

Type	Parameters	Description
uint32_t	cp_feature	CP feature value - Not supposed to be modified during the lifetime of the device
uint8_t	prfl_config	Profile characteristic configuration bit field: **Bit 0**: Enables Measurement broadcaster mode **Bit 1**: Enables Control Point Characteristic if: - Server supports configurable settings - Server can be requested for parameters
uint8_t	sensor_loc	Sensor location is stored in the environment and can be changed using the appropriate control point procedure

Response: CPPS_CMP_EVT

Description: This message shall be used to add an instance of the Cycling Power Service.

The CP Control Point characteristic will be automatically added if at least one of the following features is supported:

- Wheel Revolution Data
- Multiple Sensor Locations
- Offset compensation

2.2.2 CPPS_ENABLE_REQ

Source: TASK_APP

Destination: TASK_CPPS

Required State: IDLE

Parameters:

Type	Parameters	Description
uint8_t	conidx	Connection index
uint16_t	prfl_ntf_ind_cfg	Characteristic Configuration Descriptor bit field value for a bonded device: **Bit 0**: Measurement Characteristic client configuration **Bit 1**: Measurement Characteristic server configuration **Bit 2**: Vector Characteristic notification configuration **Bit 3**: Control Point Characteristic indication configuration

Response: CPPS_ENABLE_RSP

Description: This message shall be used after the connection with a peer in order to restore the CPP Sensor bond data

2.2.3 CPPS_ENABLE_RSP

Source: TASK_CPPS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	conidx	Connection index
uint8_t	status	Status of the operation

Description: This message corresponds to the response of setting bond data operation.

2.2.4 CPPS_GET_ADV_DATA_REQ

Source: TASK_APP

Destination: TASK_CPPS

Required State: IDLE

Parameters:

Type	Parameters	Description
struct **cpp_cp_meas**	parameters	Structure containing measurement notification fields

Description: Packs provided data in a single array including the advertising header required for the Cycling Power Service. Note the procedure does not broadcast the data; the application should implement it using the GAP interface

2.2.5 CPPS_GET_ADV_DATA_RSP

Source: TASK_CPPS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	status	Status of the operation
uint8_t	data_len	Length of the data
uint8_t array	adv_data	Array with the data

Description: Returns packed data to be broadcasted by the application.

2.2.6 CPPS_NTF_CP_MEAS_REQ

Source: TASK_APP

Destination: TASK_CPPS

Required State: IDLE

Parameters:

Type	Parameters	Description
struct **cpp_cp_meas**	parameters	Structure containing measurement notification fields

Description: This message shall be used by the application to send a CP Measurement notification to every connected device. The profile checks whether the peer device has enabled the sending of notifications in the client configuration descriptor and sends it or not depending on its value.

Note that the options present in the flags field may not fit to the link/negotiated MTU (default MTU is 23 bytes), in this case, the procedure will automatically split the notification into separate ATT PDU notifications and would be sent in succession. The split of notifications is inefficient (waste of extra ATT PDU processing), and highly discouraged. The user can avoid this by proposing a higher link MTU to the peer via GATT MTU exchange procedure.

The profile fully supports the broadcasting mode for the Measurement characteristic; nevertheless it is necessary to write “enabled” the server configuration descriptor in order to start advertising. The data should comply with broadcast mode (with non-connectable LE packets, ADV Flags properly set to Broadcast).

2.2.7 CPPS_NTF_CP_MEAS_RSP

Source: TASK_CPPS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	status	Status of the operation

Description: Message to inform the application about the status of the measurement notification.

2.2.8 CPPS_NTF_CP_VECTOR_REQ

Source: TASK_APP

Destination: TASK_CPPS

Required State: CONNECTED

Parameters:

Type	Parameters	Description
struct **cpp_cp_vector**	parameters	Structure containing vector notification fields

Description: This message shall be used by the application to send a CP Vector notification to every connected device.

The profile checks whether the peer device has enabled the sending of notifications for the characteristic and sends it or not depending on its client configuration descriptor.

2.2.9 CPPS_NTF_CP_VECTOR_RSP

Source: TASK_CPPS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	status	Status of the operation

Description: Message corresponding to the response of the vector notification.

2.2.10 CPPS_CTNL_PT_REQ_IND

Source: TASK_CPPS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint16_t	conidx	Connection index
uint8_t	op_code	Operation code (see Table in 5)
union	value	
uint32_t	cumul_value	Cumulative cycling power value
uint8_t	sensor_location	Set sensor location
uint16_t	crank_length	Value of the crank length
uint16_t	chain_length	Value of the chain length
uint16_t	chain_weight	Value of the chain weight
uint16_t	span_length	Value of the span length
uint16_t	mask_content	Measurement content mask

Description: The message is sent to the application when the CP Control Point characteristic is written by the peer device. The application shall answer using the CPPS_CTNL_PT_CFM message.

2.2.11 CPPS_CTNL_PT_CFM

Source: TASK_CPPS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint16_t	con_idx	Connection index
uint8_t	op_code	Operation code (see Table in 5)
uint8_t	status	Status (see Table in 5)
union	value	
uint32_t	supp_sensor_loc;	Supported sensor locations
uint32_t	cumul_wheel_rev	Cumulative Wheel revolution value
uint16_t	crank_length	Crank length
uint16_t	chain_length	Chain length

uint16_t	chain_weight	Chain weight
uint16_t	span_length	Span length
int16_t	offset_comp	Offset compensation
uint16_t	mask_meas_content	Mask measurement content
uint8_t	sampling_rate	Sampling rate
uint8_t	sensor_loc	New sensor location
struct **prf_date_time**	factory_calibration	Calibration date

Description: This message is sent by the application in response to the CPPS_CTNL_PT_REQ_IND message. It contains the value requested by the profile.

In the case when this message is received while no request message had been sent, it will be automatically dropped.

2.2.12 CPPS_CFG_NTFIND_IND

Source: TASK_CPPS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint16_t	conidx	Connection index
uint8_t	char_code	Descriptor code bit field: • **Bit 0**: CPP_PRF_CFG_FLAG_CP_MEAS_NTF • **Bit 1**: CPP_PRF_CFG_FLAG_SP_MEAS_NTF • ~~**Bit 2**: CPP_PRF_CFG_FLAG_VECTOR_NTF~~ • **Bit 3**: CPP_PRF_CFG_FLAG_CTNL_PT_IND
uint16_t	ntf_cfg	Notification configuration new value

Description: This message is sent to the application each time a peer device has successfully written to the Client Characteristic Configuration descriptor of the CP Measurement (client and server) or the CP Control Point characteristics.

Note: When a collector writes to the Vector Client Characteristic Configuration descriptor to start the notifications, the sensor may request new connection parameters (e.g. using the GAP Connection Parameter Update procedure) before the notifications are sent, if the current connection parameters do not allow the sending of notification (e.g. the sensor requires faster connection interval). The sensor shall use CPPS_VECTOR_CFG_CFM message in order to confirm (PRF_ERR_OK) this change or not according to the requirements within a period of time defined by the sensor.

2.2.13 CPPS_VECTOR_CFG_REQ_IND

Source: TASK_CPPS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint16_t	conidx	Connection index
uint8_t	char_code	Descriptor code bit field: • ~~**Bit 0**: CPP_PRF_CFG_FLAG_CP_MEAS_NTF~~ • ~~**Bit 1**: CPP_PRF_CFG_FLAG_SP_MEAS_NTF~~ • **Bit 2**: CPP_PRF_CFG_FLAG_VECTOR_NTF • ~~**Bit 3**: CPP_PRF_CFG_FLAG_CTNL_PT_IND~~
uint16_t	ntf_cfg	Notification configuration new value

Description: This message is sent to the application each time a peer device has successfully written to the Client Characteristic Configuration descriptor of the Vector characteristic.

Note: When a collector writes to the Vector Client Characteristic Configuration descriptor to start the notifications, the sensor may request new connection parameters (e.g. using the GAP Connection Parameter Update procedure) before the notifications are sent, if the current connection parameters do not allow the sending of notification (e.g. the sensor requires faster connection interval). The sensor shall use CPPS_VECTOR_CFG_CFM message in order to confirm (PRF_ERR_OK) this change or not according to the requirements within a period of time defined by the sensor.

2.2.14 CPPS_VECTOR_CFG_CFM

Source: TASK_APP

Destination: TASK_CPPS

Parameters:

Type	Parameters	Description
uint16_t	conidx	Connection index
uint8_t	status	Status of the operation: • PRF_ERR_OK (0x00) • Other error code value (except 0x00)
uint16_t	ntf_cfg	Notification configuration new value

Description: This message is sent by the application each time a peer device writes the Vector Client Characteristic Configuration descriptor. If there is no change of the connection parameters within a period of time defined by the application, it should return PRF_ERROR_OK in order to be able to start notifications; otherwise, it should check the provided connection parameters and response according to the validity of them.

2.2.15 CPPS_CMP_EVT

Source: TASK_CPPS

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint16_t	conidx	Connection index
uint8_t	operation	Operation code: • CPPS_NTF_MEAS_OP_CODE • CPPS_NTF_VECTOR_OP_CODE • CPPS_CTNL_PT_SET_CUMUL_VAL_OP_CODE • CPPS_CTNL_PT_UPD_SENSOR_LOC_OP_CODE • CPPS_CTNL_PT_REQ_SUPP_SENSOR_LOC_OP_CODE • CPPS_CTNL_PT_SET_CRANK_LENGTH_OP_CODE • CPPS_CTNL_PT_REQ_CRANK_LENGTH_OP_CODE • CPPS_CTNL_PT_SET_CHAIN_LENGTH_OP_CODE • CPPS_CTNL_PT_REQ_CHAIN_LENGTH_OP_CODE • CPPS_CTNL_PT_SET_CHAIN_WEIGHT_OP_CODE • CPPS_CTNL_PT_REQ_CHAIN_WEIGHT_OP_CODE • CPPS_CTNL_PT_SET_SPAN_LENGTH_OP_CODE • CPPS_CTNL_PT_REQ_SPAN_LENGTH_OP_CODE • CPPS_CTNL_PT_START_OFFSET_COMP_OP_CODE • CPPS_CTNL_MASK_CP_MEAS_CH_CONTENT_OP_CODE • CPPS_CTNL_REQ_SAMPLING_RATE_OP_CODE • CPPS_CTNL_REQ_FACTORY_CALIBRATION_DATE_OP_CODE • CPPS_CTNL_ERR_IND_OP_CODE
uint8_t	status	Status of the operation

Description: The message is used by the CPPS task to inform the sender of a command that the procedure is over and contains the status of the procedure.

3 CPP Collector Role API

3.1 Environment

Within the CPPC task, four states are defined: **FREE, IDLE, DISCOVERYING, BUSY**

Important Note: The TASK_CPPC task is multi-instantiated, an instance is created for each connection for which the profile will be enabled and each of these instances will have a different task ID. Thus, it is very important for the application to keep the source task ID of the first received CPPC_CMP_EVT message to be able to communicate with the peer device linked to this task ID once it has been enabled.

The term TASK_CPPC_IDX will be used in the rest of the document to refer to any instance of the Cycling Power profile Collector Role Task. The term TASK_CPPC will refer to the first instance of this task.

3.2 API Messages

3.2.1 Initialization

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

3.2.2 CPPC_ENABLE_REQ

Source: TASK_APP

Destination: TASK_CPPC

Required State: FREE

Parameters:

Type	Parameters	Description
uint8_t	con_type	Connection type
struct **cppc_cps_content**	cps	Service structure discovered in the database of the peer device

Response: CPPC_ENABLE_RSP

Description: This message is used for enabling the Collector role of the CPP.

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. Application shall save this information to reuse them for other connections. During normal connection, previously discovered device information can be reused.

For a normal connection, the response to this request is sent right away after saving the CPS content in the environment and registering CPPC in GATT to receive the notifications for the known attribute handles in CPS that would be notified.

For a discovery connection, discovery of the peer CPS is started and the response will be sent at the end of the discovery with the discovered attribute details.

3.2.3 CPPC_ENABLE_RSP

Source: TASK_CPPC

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	status	Status of the operation
struct **cppc_cps_content**	cps	Service structure discovered in the database of the peer device

Description: This message informs the application about the status of the enabling procedure.

3.2.4 CPPC_READ_CMD

Source: TASK_APP

Destination: TASK_CPPC_IDX

Required State: IDLE

Parameters:

Type	Parameters	Description
uint8_t	operation	Operation code set by the profile task
uint8_t	read_code	Read code: • CPPC_RD_CP_FEAT • CPPC_RD_SENSOR_LOC • CPPC_RD_WR_CP_MEAS_CL_CFG • CPPC_RD_WR_CP_MEAS_SV_CFG • CPPC_RD_WR_VECTOR_CFG • CPPC_RD_WR_CTNL_PT_CFG

Response: CPPC_VALUE_IND and CPPC_CMP_EVT

Description: The message shall be used to read the value of an attribute in the peer device database.

3.2.5 CPPC_CFG_NTFIND_CMD

Source: TASK_APP

Destination: TASK_CPPC_IDX

Required State: IDLE

Parameters:

Type	Parameters	Description
uint8_t	operation	Operation code
uint8_t	desc_code	Descriptor code: • CPPC_RD_WR_CP_MEAS_CL_CFG • CPPC_RD_WR_CP_MEAS_SV_CFG • CPPC_RD_WR_VECTOR_CFG • CPPC_RD_WR_CTNL_PT_CFG
uint16_t	ntfind_cfg	NTF/IND configuration

Response: CPPC_CMP_EVT

Description: This message is used to configure sending of notification/indication in the peer device database.

3.2.6 CPPC_CTNL_PT_CFG_REQ

Source: TASK_APP

Destination: TASK_CPPC_IDX

Required State: IDLE

Parameters:

Type	Parameters	Description
uint8_t	operation	Operation code
struct **cpp_ctnl_pt_req**	ctnl_pt	Control point request

Description: This message allows writing the value of the CP Control Point characteristic.

If the CP Control Point characteristic has not been found in the peer device database during the discovery procedure, a CPPC_CMP_EVT message is sent back to the requester with a PRF_ERR_INEXISTENT_HDL error status.

3.2.7 CPPC_CTNL_PT_RSP

Source: TASK_CPPC_IDX

Destination: TASK_APP

Parameters:

Type	Parameters	Description
struct **cpp_ctnl_pt_rsp**	rsp	Control point response structure

Description: This message is sent to the application when a new value is received from the CP control Point indication

3.2.8 CPPC_VALUE_IND

Source: TASK_CPPC_IDX

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	att_code	Attribute code: • CPPC_NTF_CP_MEAS • CPPC_NTF_CP_VECTOR • CPPC_RD_CP_FEAT • CPPC_RD_SENSOR_LOC • CPPC_RD_WR_CP_MEAS_CL_CFG • CPPC_RD_WR_CP_MEAS_SV_CFG • CPPC_RD_WR_VECTOR_CFG • CPPC_RD_WR_CTNL_PT_CFG
union	value	
struct **cpp_cp_meas**	cp_meas	CP measurement
uint32_t	sensor_feat	CP sensor feature
uint8_t	sensor_loc	Set sensor location
struct **cpp_cp_vector**	cp_vector	CP vector
uint16_t	ntf_cfg	Client characteristic configuration descriptor value

Description: This message is sent to the application when a new value is received from the peer device within a read response or a notification.

3.2.9 CPPC_CMP_EVT

Source: TASK_CPPC_IDX

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	operation	Operation code: • CPPC_READ_OP_CODE • CPPC_CFG_NTF_IND_OP_CODE • CPPC_CTNL_PT_CFG_WR_OP_CODE • CPPC_CTNL_PT_CFG_IND_OP_CODE
uint8_t	status	Status of the operation

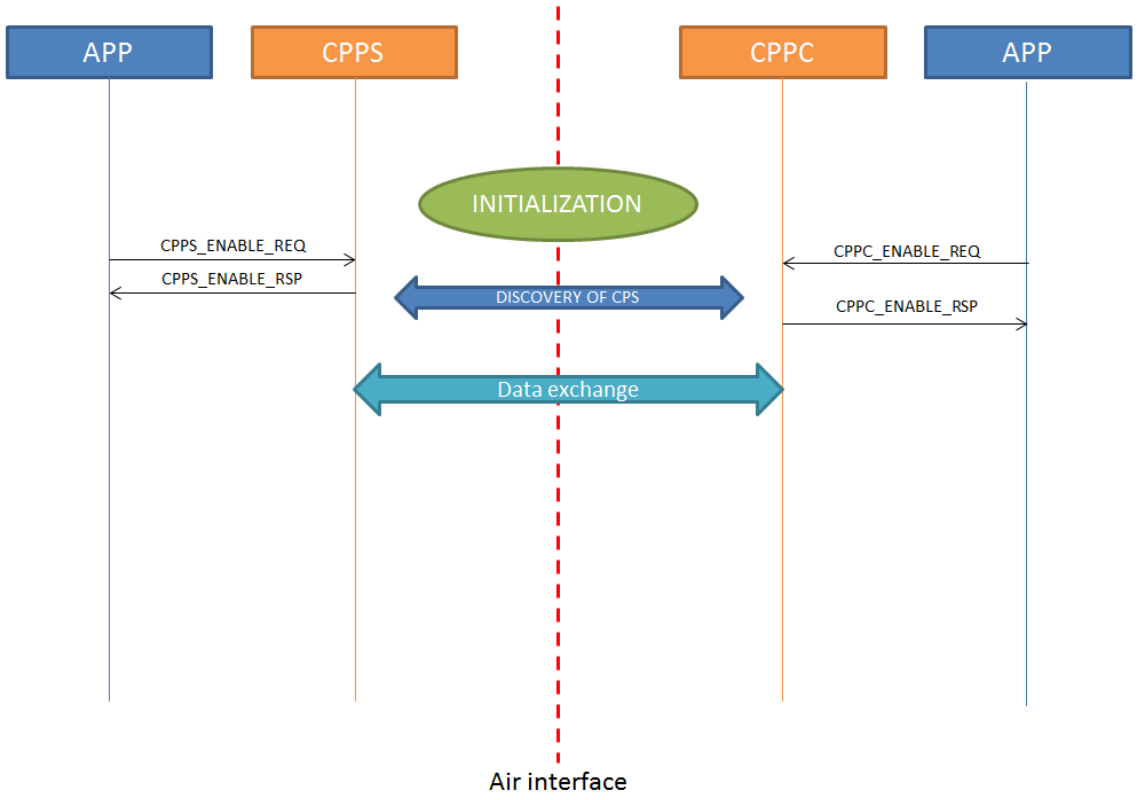
Description: The message is used by the CPPC task to inform the sender of a command that the procedure is over and contains the status of the procedure.

4 Message Sequence Charts (MSCs)

This part describes the different procedures that can be used within the CPP.

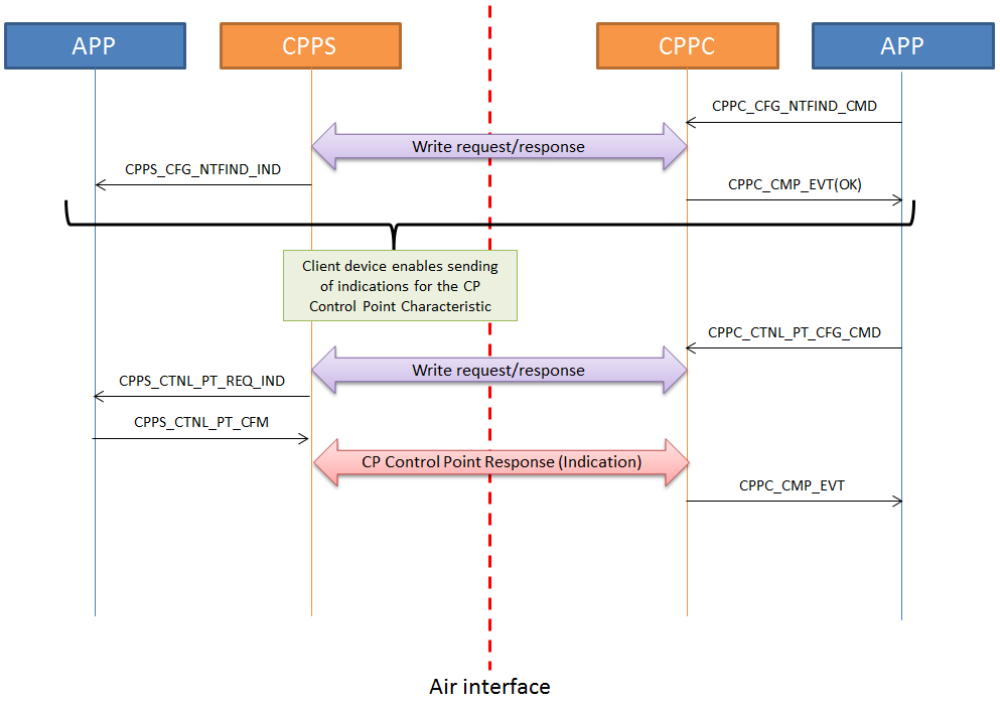
In these MSCs, it is assumed that the two RW stacks (one with the sensor role and another one with the collector role) are connected together and both tasks created and allocated using GAPM procedures.

4.1 Device Initialization / Connection / Disconnection



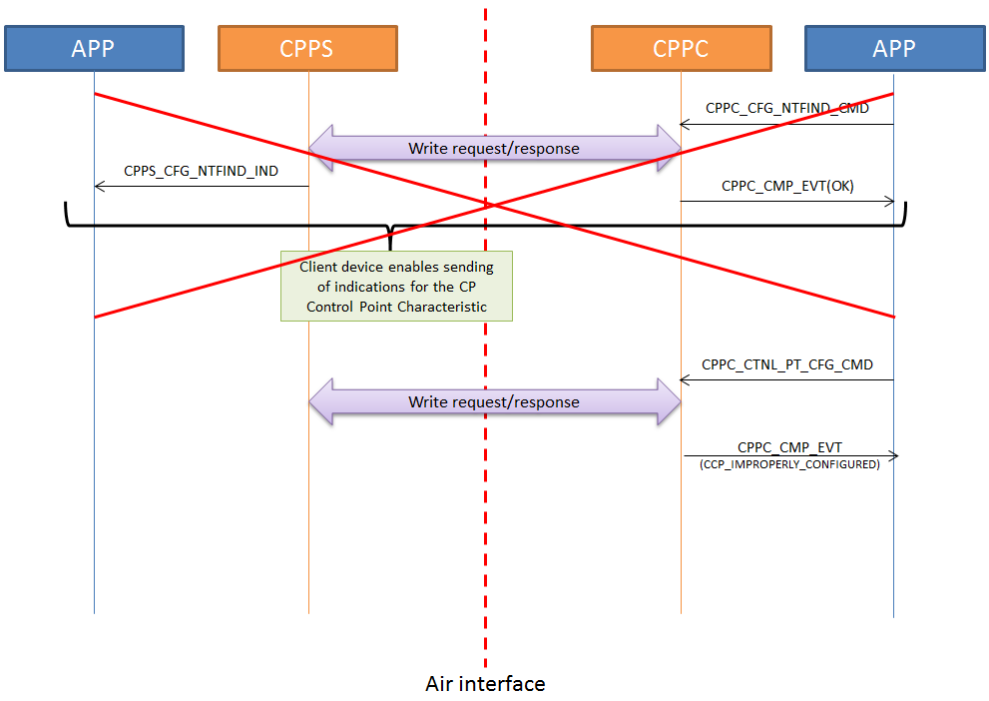
4.2 CP Control Point Characteristic usage

4.2.1 Normal Procedure



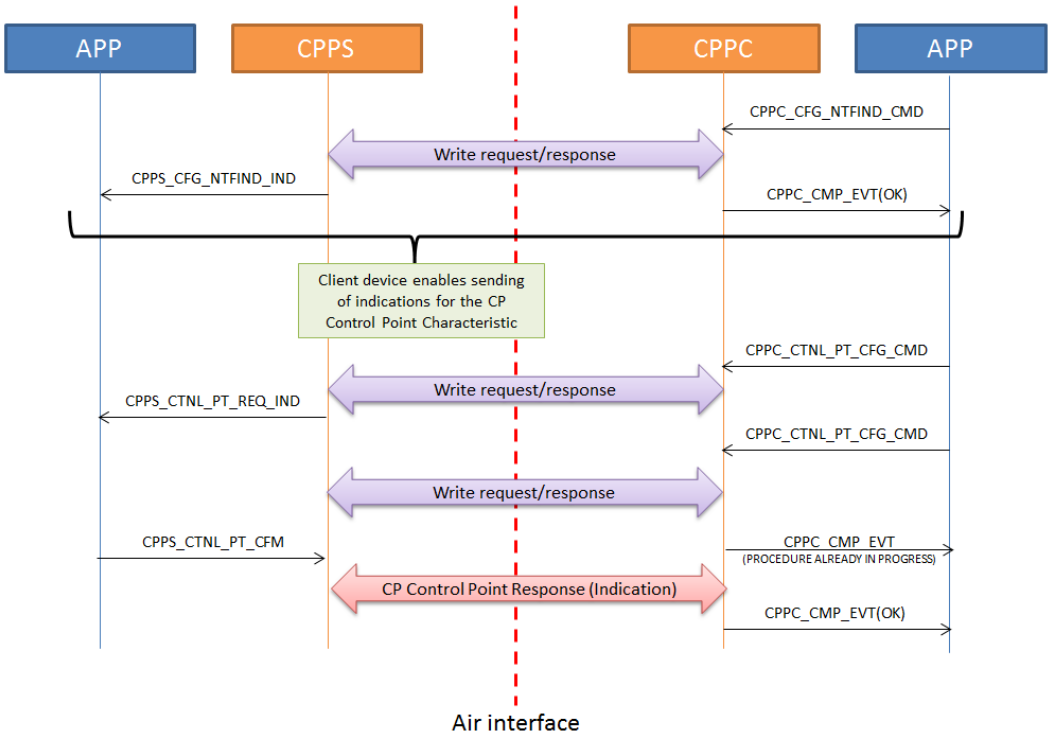
4.2.2 CPP Improperly Configured Error

If the client device has not enabled the sending of indications to the peer, the server device will answer with a **CPP_ERROR_IMPROPERLY_CONFIGURED** error.



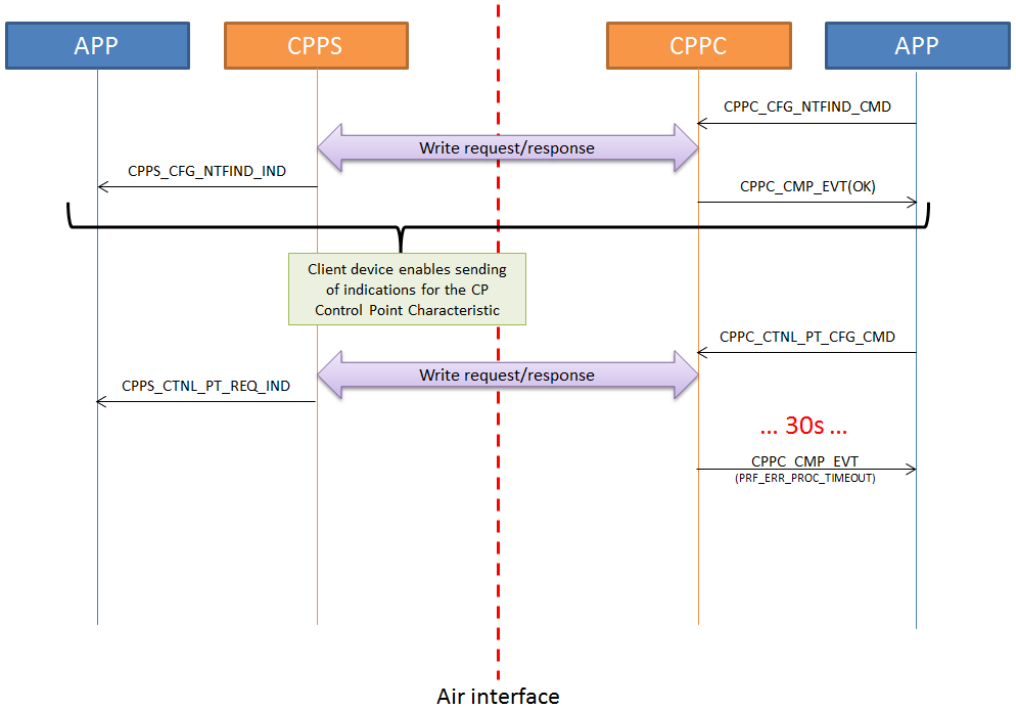
4.2.3 Procedure Already in Progress Error

If the client device writes the Control Point characteristic while the previous procedure is not over, the server will answer with a **PROCEDURE_ALREADY_IN_PROGRESS** error.



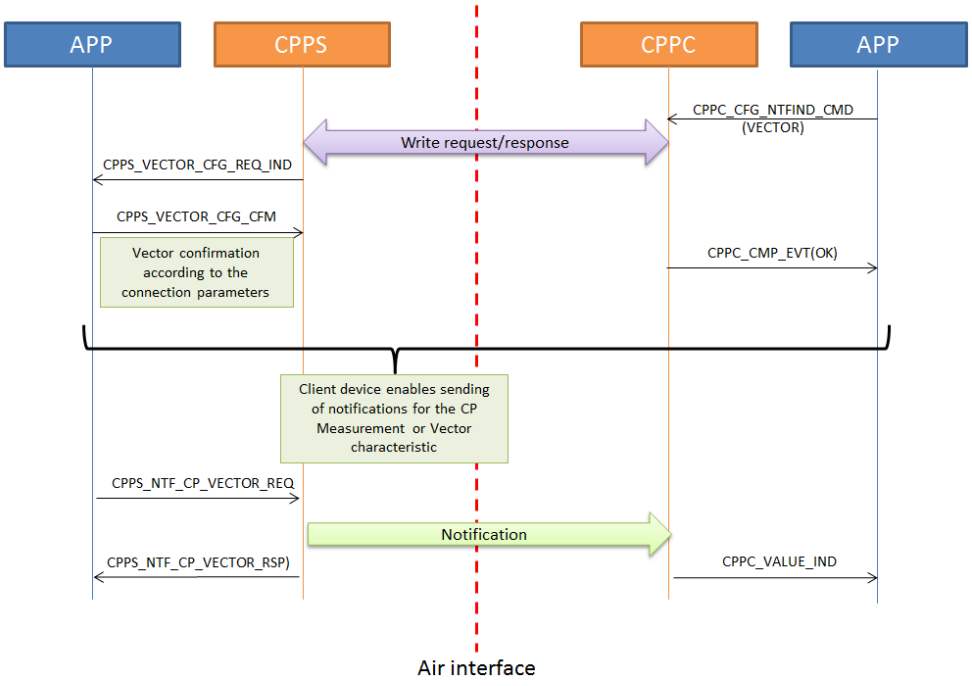
4.2.4 Procedure Timeout

If the client device does not receive a Control Point response within 30s after reception of the write response, a procedure timeout error will be raised.



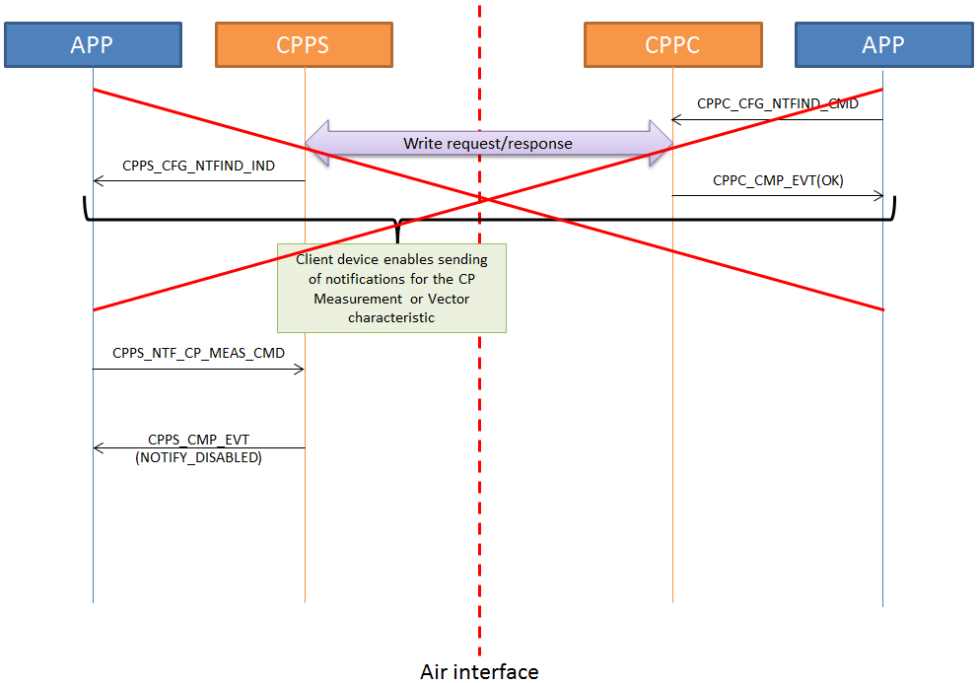
4.3 Sending / Receiving of CP Measurements and Vector

4.3.1 Normal Procedure



4.3.2 Sending of Notifications Disabled

If sending of notifications has not been enabled by the collector device, the server device won't be able to send measurements, a **PRF_ERR_NOTIFY_DISABLED** error will be sent back to the application.



5 Miscellaneous

Name	Value	Description
CPP_MEAS_PEDAL_POWER_BALANCE_PRESENT	0x0001	Pedal Power Balance Present
CPP_MEAS_PEDAL_POWER_BALANCE_REFERENCE	0x0002	Pedal Power Balance Reference
CPP_MEAS_ACCUM_TORQUE_PRESENT	0x0004	Accumulated Torque Present
CPP_MEAS_ACCUM_TORQUE_SOURCE	0x0008	Accumulated Torque Source
CPP_MEAS_WHEEL_REV_DATA_PRESENT	0x0010	Wheel Revolution Data Present
CPP_MEAS_CRANK_REV_DATA_PRESENT	0x0020	Crank Revolution Data Present
CPP_MEAS_EXTREME_FORCE_MAGNITUDES_PRESENT	0x0040	Extreme Force Magnitudes Present
CPP_MEAS_EXTREME_TORQUE_MAGNITUDES_PRESENT	0x0080	Extreme Torque Magnitudes Present
CPP_MEAS_EXTREME_ANGLES_PRESENT	0x0100	Extreme Angles Present
CPP_MEAS_TOP_DEAD_SPOT_ANGLE_PRESENT	0x0200	Top Dead Spot Angle Present
CPP_MEAS_BOTTOM_DEAD_SPOT_ANGLE_PRESENT	0x0400	Bottom Dead Spot Angle Present
CPP_MEAS_ACCUM_ENERGY_PRESENT	0x0800	Accumulated Energy Present
CPP_MEAS_OFFSET_COMPENSATION_INDICATOR	0x1000	Offset Compensation Indicator

Table 1 – CP Measurement Present Parameters bit flags

Name	Value	Description
CPP_FEAT_PEDAL_POWER_BALANCE_SUPP	0x00000001	Pedal Power Balance Supported
CPP_FEAT_ACCUM_TORQUE_SUPP	0x00000002	Accumulated Torque Supported
CPP_FEAT_WHEEL_REV_DATA_SUPP	0x00000004	Wheel Revolution Data Supported
CPP_FEAT_CRANK_REV_DATA_SUPP	0x00000008	Crank Revolution Data Supported
CPP_FEAT_EXTREME_MAGNITUDES_SUPP	0x00000010	Extreme Magnitudes Supported
CPP_FEAT_EXTREME_ANGLES_SUPP	0x00000020	Extreme Angles Supported
CPP_FEAT_TOPBOT_DEAD_SPOT_ANGLES_SUPP	0x00000040	Top and Bottom Dead Spot Angles Supported
CPP_FEAT_ACCUM_ENERGY_SUPP	0x00000080	Accumulated Energy Supported
CPP_FEAT_OFFSET_COMP_IND_SUPP	0x00000100	Offset Compensation Indicator Supported
CPP_FEAT_OFFSET_COMP_SUPP	0x00000200	Offset Compensation Supported
CPP_FEAT_CP_MEAS_CH_CONTENT_MASKING_SUPP	0x00000400	CP Measurement CH Content Masking Supported
CPP_FEAT_MULT_SENSOR_LOC_SUPP	0x00000800	Multiple Sensor Locations Supported
CPP_FEAT_CRANK_LENGTH_ADJ_SUPP	0x00001000	Crank Length Adjustment Supported
CPP_FEAT_CHAIN_LENGTH_ADJ_SUPP	0x00002000	Chain Length Adjustment Supported
CPP_FEAT_CHAIN_WEIGHT_ADJ_SUPP	0x00004000	Chain Weight Adjustment Supported
CPP_FEAT_SPAN_LENGTH_ADJ_SUPP	0x00008000	Span Length Adjustment Supported
CPP_FEAT_SENSOR_MEAS_CONTEXT	0x00010000	Sensor Measurement Context
CPP_FEAT_INSTANT_MEAS_DIRECTION_SUPP	0x00020000	Instantaneous Measurement Direction Supported
CPP_FEAT_FACTORY_CALIBRATION_DATE_SUPP	0x00040000	Factory Calibration Date Supported

Table 2 – CP Feature supported Parameters bit flags

Name	Value	Description
CPP_LOC_OTHER	0	Other
CPP_LOC_TOP_SHOE	1	Top of shoe
CPP_LOC_IN_SHOE	2	In shoe
CPP_LOC_HIP	3	Hip
CPP_LOC_FRONT_WHEEL	4	Front Wheel
CPP_LOC_LEFT_CRANK	5	Left Crank
CPP_LOC_RIGHT_CRANK	6	Right Crank
CPP_LOC_LEFT_PEDAL	7	Left Pedal
CPP_LOC_RIGHT_PEDAL	8	Right Pedal
CPP_LOC_FRONT_HUB	9	Front Hub
CPP_LOC_REAR_DROPOUT	10	Rear Dropout
CPP_LOC_CHAINSTAY	11	Chain stay
CPP_LOC_REAR_WHEEL	12	Rear Wheel
CPP_LOC_REAR_HUB	13	Rear Hub
CPP_LOC_CHEST	14	Chest

Table 3 – Sensor Location Keys

Name	Value	Description
CPP_VECTOR_CRANK_REV_DATA_PRESENT	0x01	Crank Revolution Data Present
CPP_VECTOR_FIRST_CRANK_MEAS_ANGLE_PRESENT	0x02	First Crank Measurement Angle Present
CPP_VECTOR_INST_FORCE_MAGNITUDE_ARRAY_PRESENT	0x04	Instantaneous Force Magnitude Array Present
CPP_VECTOR_INST_TORQUE_MAGNITUDE_ARRAY_PRESENT	0x08	Instantaneous Torque Magnitude Array Present
CPP_VECTOR_INST_MEAS_DIRECTION_LSB	0x10	Instantaneous Measurement Direction LSB
MSB CPP_VECTOR_INST_MEAS_DIRECTION_MSB	0x20	Instantaneous Measurement Direction

Table 4 – CP Vector flags

Name	Value	Description
CPP_CTNL_PT_SET_CUMUL_VAL	1	Set Cumulative Value
CPP_CTNL_PT_UPD_SENSOR_LOC	2	Update Sensor Location
CPP_CTNL_PT_REQ_SUPP_SENSOR_LOC	3	Request Supported Sensor Locations
CPP_CTNL_PT_SET_CRANK_LENGTH	4	Set Crank Length
CPP_CTNL_PT_REQ_CRANK_LENGTH	5	Request Crank Length
CPP_CTNL_PT_SET_CHAIN_LENGTH	6	Set Chain Length
CPP_CTNL_PT_REQ_CHAIN_LENGTH	7	Request Chain Length
CPP_CTNL_PT_SET_CHAIN_WEIGHT	8	Set Chain Weight
CPP_CTNL_PT_REQ_CHAIN_WEIGHT	9	Request Chain Weight
CPP_CTNL_PT_SET_SPAN_LENGTH	10	Set Span Length
CPP_CTNL_PT_REQ_SPAN_LENGTH	11	Request Span Length
CPP_CTNL_PT_START_OFFSET_COMP	12	Start Offset Compensation
CPP_CTNL_MASK_CP_MEAS_CH_CONTENT	13	Mask CP Measurement Characteristic Content
CPP_CTNL_REQ_SAMPLING_RATE	14	Request Sampling Rate
CPP_CTNL_REQ_FACTORY_CALIBRATION_DATE	15	Request Factory Calibration Date
CPP_CTNL_PT_RSP_CODE	32	Response Code

Table 5 – CP Control Point Operation Code Keys

Name	Value	Description
CPP_CTNL_PT_RESP_SUCCESS	1	Success
CPP_CTNL_PT_RESP_NOT_SUPP	2	Operation Code Not Supported
CPP_CTNL_PT_RESP_INV_PARAM	3	Invalid Parameter
CPP_CTNL_PT_RESP_FAILED	4	Operation Failed

Table 6 – CP Control Point Response Value Keys

Type	Parameters	Description
uint16_t	flags	Flags
int16_t	inst_power	Instantaneous Power
uint8_t	pedal_power_balance	Pedal Power Balance
uint16_t	accum_torque	Accumulated torque
uint32_t	cumul_wheel_rev	Cumulative Wheel Revolutions
uint16_t	last_wheel_evt_time	Last Wheel Event Time
uint16_t	cumul_crank_rev	Cumulative Crank Revolution
uint16_t	last_crank_evt_time	Last Crank Event Time
int16_t	max_force_magnitude	Maximum Force Magnitude
int16_t	min_force_magnitude	Minimum Force Magnitude
int16_t	max_torque_magnitude	Maximum Torque Magnitude
int16_t	min_torque_magnitude	Minimum Torque Magnitude
uint16_t	max_angle	Maximum Angle (12 bits)
uint16_t	min_angle	Minimum Angle (12bits)
uint16_t	top_dead_spot_angle	Top Dead Spot Angle
uint16_t	bot_dead_spot_angle	Bottom Dead Spot Angle
uint16_t	accum_energy	Accumulated energy

Table 7 – CP Measurement Structure (struct cpp_cp_meas)

Type	Parameters	Description
uint8_t	flags	Flags
uint8_t	nb	Force-Torque Magnitude Array Length
uint16_t	cumul_crank_rev	Cumulative Crank Revolutions
uint16_t	last_crank_evt_time	Last Crank Event Time
uint16_t	first_crank_meas_angle	First Crank Measurement Angle
int16_t [_ARRAY_EMPTY]	force_torque_magnitude	Mutually excluded Force and Torque Magnitude Arrays

Table 8 – CP Vector Structure (struct cpp_cp_vector)

Type	Parameters	Description
uint16_t	year	Year time element
uint8_t	month	Month time element
uint8_t	day	Day time element
uint8_t	hour	Hour time element
uint8_t	min	Minute element
uint8_t	sec	Second element

Table 9 – Date time Structure (struct prf_date_time)

6 Abbreviations

Abbreviation	Original Terminology
API	Application Programming Interface
BLE	Bluetooth Low Energy
GAP	Generic Access Profile
GATT	Generic Attribute Profile
CPP	Cycling Power Profile
CPPS	Cycling Power Sensor Role
CPPC	Cycling Power Collector Role
CPS	Cycling Power Service
MSC	Message Sequence Chart
RW	RivieraWaves SAS
SM	Security Manager

7 References

[1]	**Title**	CYCLING POWER PROFILE SPECIFICATION		
	Reference	CPP_SPEC_V10		
	Version	V10r00	**Date**	2012-30-04
	Source	Bluetooth SIG		

[2]	**Title**	CYCLING POWER PROFILE TEST SPECIFICATION		
	Reference	CPP.TS.1.0.0		
	Version	1.0.0	**Date**	2012-30-04
	Source	Bluetooth SIG		

[3]	**Title**	CYCLING POWER SERVICE SPECIFICATION		
	Reference	CPS_SPEC_V10		
	Version	V10r00	**Date**	2012-30-04
	Source	Bluetooth SIG		

[4]	**Title**	CYCLING POWER SERVICE TEST SPECIFICATION		
	Reference	CPS.TS.1.0.0		
	Version	1.0.0	**Date**	2012-30-04
	Source	Bluetooth SIG		