SentecLink Protocol

SMI Mode/SentecLink Online Mode

1. Introduction

The SentecLink Protocol supports two modes that do co-exist on the serial communication link. As default, the simple SentecLink Online Mode is active (e.g. after switching on the SDM). The bidirectional SMI Mode extends the SentecLink Online Mode.

Note: When using LAN communication link, only SMI Mode is supported for communication.

To use SMI commands via serial communication link, the 'SentecLink Protocol' must be set in the menu of the Sentec Digital Monitor (SDM). Selectable baud rates are 19'200, 38'400, 57'600 and 115'200, which is the default baud rate.

SentecLink Online Mode starts communication automatically if the SDM is interfaced via serial communication link. Switch off the online output by sending '/online=off<CR><LF>' (**note** that online mode will automatically restart after 2o minutes if no commands are sent by the client.

Note: The bidirectional SMI Mode and the communication protocol (ASCII) of the upcoming "Sentec SensorLink" (Sentec's OEM Solution) are equivalent, although data objects may be subject to changes.

2. Reference documents

Technical Manual for the SDM (HB-005752 SDM_Technical_Manual)
Instruction Manual for the SDM (HB-005771 SDMS_InstructionManual_EN)
RF006398-SMB_SMI_Objects_r4211.docx

The latest version of the Technical Manual and Instructions for use for the SDM can be downloaded from https://www.sentec.com/ifu/.

This document is based on software version SMB V08.04 as well as on the documents listed above.

3. General Aspects

Aspects to consider when developing an SMI Mode based Sentec Driver:

1) SMI Mode can be used via Serial Link or via LAN connection.

2) We recommend that you implement all primary parameters (tcPCO2, tcPO2, SpO2, PR) and all secondary parameters (PI, HP) supported by the current SMB version (see section 2).

   Note: If the SDM is operated in neonatal mode, SpO2, PR, PI are not supported.

3) Various parameters are flagged with a quality. Possible qualities are "valid, questionable, unstable, not valid, not available". Parameters should only be displayed if the quality is "valid" or "questionable". If "questionable" we recommend to display a "?" (question mark) adjacent to the value.
   Example: The SMI Mode unfortunately outputs tcPCO2 and tcPO2 values when the sensor is in the Docking Station (Calibration Chamber). Obviously, in this situation tcPCO2 and tcPO2 data correspond to technical data, so the values are flagged as invalid and have to be ignored.

   Note: If a parameter is flagged to be "unstable", you may – as implemented on the SDM - display it in dimmed grey style.
   If a parameter is flagged to be "questionable", you may – as implemented on the SDM – display it with a question mark adjacent the value.
   If a parameter is flagged to be "invalid", you may – as implemented on the SDM – replace the values by "---".
   If a parameter is flagged to be "not available", you may – as implemented on the SDM – replace the values by "-/-".

4) We recommend to duplicate/store the SDM's alarm limits and conditions on the host system.

   Note: It is optionally possible to change the SDM's alarm settings from the host system.

5) We recommend to implement retrospective correction of residual PCO2 Drift and of residual PO2 Drift (requires use of SMI Mode).

6) On the SDM, units for tcPCO2 and tcPO2 can be set to "mmHg" or to "kPa". We recommend that you offer this possibility on the host system as well.

   Note: Regardless on the PCO2/PO2 unit that is active on the SDM, tcPCO2 and tcPO2 are always presented in "mmHg" when SMI Mode is used.

   Note: 1 kPa corresponds to 7.5 mmHg

7) We recommend that you store the Status Codes as well as all relevant system information for documentation purposes (selected patient type, SpO2 Averaging, selected Sensor SET temperature as well as serial numbers/software versions of the SDM and connected sensor).

   Note: A list of all Status Codes with explanations can be found in the current technical manual (see section 2)

   Note: It is possible that no Status Code is active.

8) If the serial protocol “SentecLink” is selected it is possible to adjust the baud rate in the menu of the SDM. Selectable options are 115’200, 57’600, 38’400, 19’200. It is therefore recommended that the host system “scans” the baud rate until communication is established.

9) Ensure to read the software version of the SDM and not to support communication with software versions older than the current SMB version (see section 2). Please ensure not to support software version SMB V09.00.x or higher.

10) If the connection between the host system and the SDM is interrupted, (e.g. cable disconnected, selection of a different Serial protocol or baud rate on the SDM, SDM switched off …) a corresponding alarm should be triggered on the host system. Once the connection is re-established, communication should restart automatically.

4. SMI Interfacing

This chapter describes how to interface with the Sentec Digital Monitor via bi-directional Sentec Monitor Interface (SMI) protocol.

4.1 Basic interfacing using SMI protocol

General Protocol Format of SMI commands

Once a serial or LAN/UDP connection has been established the bi-directional interface follows a simple request and answer scheme: a command or object is sent and the SDM answers with a confirmation and/or with data. A command or object is always terminated with carriage return (CR) and line feed (LF) or vertical tab (VT) and CRC characters (see below). All encoding is in basic ASCII format.

Example:

Send:	`<SMI-object>[=<data>]<CR><LF>`	; SMI object with optional data field
Response:	`<SMI-object>=<data><CR><LF>`	; SMI object with data response

In the following examples, the control characters will not be repeated for better legibility.

Protected SMI commands: CRC calculation

8-bit CRC: polynomial = x8 + x5 + x4 + 1 = 0x8C (reverse bit order), start value = 0

Implementation hints:

```
const BYTE cStartValue = 0;

const BYTE crc8table[256]={
0x00, 0x5e, 0xbc, 0xe2, 0x61, 0x3f, 0xdd, 0x83, 0xc2, 0x9c, 0x7e, 0x20, 0xa3, 0xfd, 0x1f, 0x41,
0x9d, 0xc3, 0x21, 0x7f, 0xfc, 0xa2, 0x40, 0x1e, 0x5f, 0x01, 0xe3, 0xbd, 0x3e, 0x60, 0x82, 0xdc,
0x23, 0x7d, 0x9f, 0xc1, 0x42, 0x1c, 0xfe, 0xa0, 0xe1, 0xbf, 0x5d, 0x03, 0x80, 0xde, 0x3c, 0x62,
0xbe, 0xe0, 0x02, 0x5c, 0xdf, 0x81, 0x63, 0x3d, 0x7c, 0x22, 0xc0, 0x9e, 0x1d, 0x43, 0xa1, 0xff,
0x46, 0x18, 0xfa, 0xa4, 0x27, 0x79, 0x9b, 0xc5, 0x84, 0xda, 0x38, 0x66, 0xe5, 0xbb, 0x59, 0x07,
0xdb, 0x85, 0x67, 0x39, 0xba, 0xe4, 0x06, 0x58, 0x19, 0x47, 0xa5, 0xfb, 0x78, 0x26, 0xc4, 0x9a,
0x65, 0x3b, 0xd9, 0x87, 0x04, 0x5a, 0xb8, 0xe6, 0xa7, 0xf9, 0x1b, 0x45, 0xc6, 0x98, 0x7a, 0x24,
0xf8, 0xa6, 0x44, 0x1a, 0x99, 0xc7, 0x25, 0x7b, 0x3a, 0x64, 0x86, 0xd8, 0x5b, 0x05, 0xe7, 0xb9,
0x8c, 0xd2, 0x30, 0x6e, 0xed, 0xb3, 0x51, 0x0f, 0x4e, 0x10, 0xf2, 0xac, 0x2f, 0x71, 0x93, 0xcd,
0x11, 0x4f, 0xad, 0xf3, 0x70, 0x2e, 0xcc, 0x92, 0xd3, 0x8d, 0x6f, 0x31, 0xb2, 0xec, 0x0e, 0x50,
0xaf, 0xf1, 0x13, 0x4d, 0xce, 0x90, 0x72, 0x2c, 0x6d, 0x33, 0xd1, 0x8f, 0x0c, 0x52, 0xb0, 0xee,
0x32, 0x6c, 0x8e, 0xd0, 0x53, 0x0d, 0xef, 0xb1, 0xf0, 0xae, 0x4c, 0x12, 0x91, 0xcf, 0x2d, 0x73,
0xca, 0x94, 0x76, 0x28, 0xab, 0xf5, 0x17, 0x49, 0x08, 0x56, 0xb4, 0xea, 0x69, 0x37, 0xd5, 0x8b,
0x57, 0x09, 0xeb, 0xb5, 0x36, 0x68, 0x8a, 0xd4, 0x95, 0xcb, 0x29, 0x77, 0xf4, 0xaa, 0x48, 0x16,
0xe9, 0xb7, 0x55, 0x0b, 0x88, 0xd6, 0x34, 0x6a, 0x2b, 0x75, 0x97, 0xc9, 0x4a, 0x14, 0xf6, 0xa8,
0x74, 0x2a, 0xc8, 0x96, 0x15, 0x4b, 0xa9, 0xf7, 0xb6, 0xe8, 0x0a, 0x54, 0xd7, 0x89, 0x6b, 0x35 };
static BYTE CRC;
```

with every new_char in SMI command (including <CR><VT> characters) do:

```
CRC = crc8table [CRC ^ new_char];          // xor
```

Append this binary value to the end of the command string.

Reset CRC before next command is processed:

```
CRC = cStartValue;
```

Initiating an SMI communication on serial link

Preparation
If SentecLink protocol is activated in the monitor settings the online data output is normally active (see SDMS Technical Manual for additional information). This means that a data line is output every second. It is recommended to disable online data output when preparing for or when running an SMI communication link:

Disable online data output (will reset by itself after approx. 20 minutes of no communication activity on serial link):

```
Send:       /online=off
Response:   /online=off
```

Initiating an SMI communication on LAN (UDP)

Only SMI command interface is supported. No specific protocol initialization steps are required; however, make sure to request SMI protocol version for subsequent version depending communication. LAN activation and IP address and port configuration of the SDM can be done via the PC software V-STATS™. Command termination with CRC is mandatory over LAN.

```
Send:       /                          ;
Response:   /SMI=xxxxxxxx,SMB,UDP      ; SMI protocol mode, x denoting protocol version, store for later use
Send:       SmiVersion                 ; different command to retrieve SMI version
Response:   SmiVersion=xxxxxxxx        ; SMI protocol version, store for later use
```

To save on network load it is recommended that the client group SMI commands in one UDP frame. Command separation is accomplished by <CR><VT><CRC>. The SDM will in turn also answer with grouped UDP frames.

4.2 Data-Pooling or Data-Streaming

The 'SMI Protocol' also supports a "Data-streaming/ Automated Notification" mode:

- **To activate** data-streaming for an object simply add ">>" behind the object-name (e.g. to activate streaming for SpO2 send "PoxSpO2>>")
- **To deactivate** data-streaming for an object simply add "<<" behind the object-name (e.g. to deactivate streaming for SpO2 send "PoxSpO2<<")
- **To switch off** all active streaming objects at the same time simply add "<<" without object-names

4.3 Qualities

Use the quality bits to derive the validity of a numerical value. Bits 0 to 3 indicate channel specific information as described in the tables below. The general quality is defined in bits 4 to 7 as follows and applies for all channels and parameters:

Valid	0x00	; value is stable
Questionable	0x10	; value is questionable, signal quality is reduced, display value e.g. with question mark
Unstable	0x20	; value is unstable, in the process of stabilization, display numerical values e.g. greyed out, do not include in trending data
Invalid	0x30	; value is invalid, do not display, use place holder such as '---', no trending
Not available	0x40	; value is not available (e.g. no sensor present or sensor does not support channel), us place holder such as '-/-', no trending

Examples:

```
Pco2Part=40.2|0   ; PCO2 is valid as general quality is 0, no special bits are flagged
Pco2Part=30.2|22  ; PCO2 is unstable as general quality is 2, then special bit 1 is set indicating that an artefact was detected
Pco2Part=48.6|8   ;  PCO2 is valid as general quality is 0, then special bit 3 is set indicating a high alarm limit condition
Pco2Part=54.3|18  ; PCO2 is questionable (reduced quality) as general quality is 1, then special bit 3 is set indicating a high alarm limit   condition
Pco2Part=28.3|4   ; PCO2 is valid as general quality is 0, then special bit 2 is set indicating a low alarm limit condition
```

5. Objects recommended to be included in an SMI Mode based Sentec Driver

When developing an SMI Mode based Sentec Driver, we recommend including the following objects.

Note: To communicate with the SDM the Name can be used or the hex values in the ID column. If the hex value is used, place an '&' in front of it, e.g. for 'AppStatus': &13. For the upcoming Sentec Sensor Link the hex values will not be available.

5.1 General System Status Information

ID	Name	Description	Type / Size	Access	Unit / Coding / Range / Example / Default	Change / Data rate
0x13	**AppStatus** -	Shows the actual sensor position	Enum 1 byte	R-,R-,R-	AppStatus=establishing 0: 'measuring' 1: 'floating' 2: 'docking' 3: 'establishing' sensor connection	Typically 1 sec (AppDisplayPeriod)
0x17	**AppPatientType**	Selects the Patient Type. Changing the patient type will re-set the Probe temperature and maximum site-time to default values. This value may change to a new Default value, when a Sensor is connected. Writing not allowed during monitoring.	Enum 1 byte	RW, RW, RW	AppPatientType=adult 0: adult 1: neonate	By menu (and AppControl)
0x16	**AppMonitoringTime**	Shows the minimum of calibration time and remaining site time.	Int 2+1 bytes	R-,R-,R-	1440…0 [min] AppMonitoringTime=0|0 Quality attribute (in hex format): Bit 7..4: general Quality Bit 3…0: reserved	1 min when > 0
0x83	**AppStatusCodes**	Shows messages represented as space separated Status Codes. Displays the actual status codes in priority sorted order; the most important code is listed first. On change of AppStatus (measuring | floating | docking | establishing) the actual status codes will change to a new set of codes.	String 1..32 bytes	R-,R-,RW	AppStatusCodes="" Space separated list of Codes	(On change by AppStatusMsg)
0x86	**AppStatusText**	Reflects messages as a default text in English Refer to "DisplayStatusText" for a translated version	String 1..96 bytes	R-,R-,RW	AppStatusText=""	(On change by AppStatusTextId)

ID	Name	Description	Type / Size	Access	Unit / Coding / Range / Example / Default	Change / Data rate
-	**DisplayStatusText** **Note:** Messages of "DisplayStatusText" are coded in Unicode that follows special rules. A Unicode character starts with a "\u" followed by 2 Bytes or with "\U" followed by the LSB. Using the MSB from the last used "\u" character. Example: DisplayStatusText = \u30b7\Ue8\Ua6\Uab\Uce\Ua6 (meaning: "Ready for use" in Japanese)	Reflects AppStatusText, translated into the current display language Reads translation from HTML file	String 1..96 bytes	R-,R-,RW	DisplayStatusText=""	(By AppStatusTextId and DisplayLanguage)
0xD2	**DisplayLanguageOption**	Generally, enables the Language selection menu This parameter can only be changed via SMI Command. If disabled the respective menu parameter is displayed in gray.	Int16 2 bytes	R-, RW,RW	0...1 DisplayLanguageOption=0 0 = disabled 1 = enabled	By external service access only
0xD3	**DisplayLanguage**	Selects the display language	Enum 1 byte	RW, RW, RW	DisplayLanguage=english 0: German 1: English 2: French 3: Italian 4: Spanish 5: Japanese 6: Swedish 7: Dutch 8: Danish 9: Portuguese 10: Turkish 11: Polish 12: Czech 14: Norwegian 15: Catalan 16: Russian 17: Chinese	By external service (or DisplayLanguageEvent)

5.2 Primary Monitoring Parameters, Pleth and important related settings

ID	Name	Description	Type / Size	Access	Unit / Coding / Range / Example / Default	Change / Data rate
0x4A	**Pco2Part** **note** the quality bits 0 to 7 attributed to this object, in particular bits 4 to 7 for the general quality Example: "Pco2Part=44.2 | 40" indicates that the PCO2 value (=44.2 mmHg) is "not available" (as quality = 40)	Shows the un-scaled pCO2 value	Float 4+1 bytes	R-,R-,RW	0...200 [mmHg] Pco2Part=0.0|40 Quality attribute (in hex format): Bit 0: IVC referenced value Bit 1: pCO2 measurement artefact detected Bit 2: Low Alarm Limit violation Bit 3: High Alarm Limit violation Bit 4..7: general Quality	Typically 1 sec (Pco2DisplayPeriod)
0x7B	**Po2** **note** the quality bits 0 to 7 attributed to this object, in particular bits 4 to 7 for the general quality	Shows the un-scaled pO2 skin value	Float 4+1 bytes	R-,R-,RW	0...200 [mmHg] Po2Part=0.0|40 Quality attribute (in hex format): Bit 0: n/a Bit 1: pO2 measurement artefact detected Bit 2: Low Alarm Limit violation Bit 3: High Alarm Limit violation Bit 4..7: general Quality	Typically 1 sec (Po2DisplayPeriod)
0x35	**PoxSpO2** **note** the quality bits 0 to 7 attributed to this object, in particular bits 4 to 7 for the general quality	Shows the SpO2 measurement value	Int 2+1 bytes	R-,R-,RW	0...100 [%] PoxSpO2=0|40 Quality attribute (in hex format): Bit 0..1: reserved Bit 2: Low Alarm Limit violation Bit 3: High Alarm Limit violation Bit 4..7: general Quality	Typically 1 sec (PoxDisplayPeriod)
0x36	**PoxPR** **note** the quality bits 0 to 7 attributed to this object, in particular bits 4 to 7 for the general quality	Shows the pulse rate measurement value	Int 2+1 bytes	R-,R-,RW	30...250 [bpm] PoxPR=0|40 Quality attribute (in hex format): Bit 0..1: reserved Bit 2: Low Alarm Limit violation Bit 3: High Alarm Limit violation Bit 4..7: general Quality	Typically 1 sec (PoxDisplayPeriod)
0x37	**PoxPI** **note** the quality bits 0 to 7 attributed to this object, in particular bits 4 to 7 for the general quality	Shows the Pulsation Index measurement value to a certain extend an indirect measure of local skin blood flow	Int 4+1 bytes	R-,R-,RW	0.00...22.00 [%] PoxPI=0.00|40 Quality attribute (in hex format): Bit 0..3: reserved Bit 4..7: general Quality	Typically 1 sec (PoxDisplayPeriod)
0x39	**PoxPleth**	Shows the actual Plethysmogram array	Int (Array)	R-,R-,R-	PoxPleth=0	(By PoxPlethDisplayPeri

ID	Name	Description	Type / Size	Access	Unit / Coding / Range / Example / Default	Change / Data rate
		The Plethysmogram values are scaled to the range -1000...+1000	1+2*n bytes		Field(0): number of array elements (decimal) Hex format: Field(1..n): Plethysmogram Value + Status Bits Bit(15): Pulse synchronization Beep Bit (14): Sample is invalid Bit (13...12): reserved Bit (11...0): signed Plethysmogram value	od)
0x33	**PoxSpO2Averaging**	Selects the averaging mode for SpO2 measurements This parameter can only be changed via SMI Command. If disabled the respective menu parameter is displayed in grey.	Int 2 bytes	RW, RW, RW	0...32 PoxSpO2Averaging=6	By menu

Pulse Plethysmogram

The pulse plethysmogram has the highest bandwidth demand when interfacing the SDM. In a fixed interval, an array of 6 values is provided for 6 plethysmogram samples. The fixed interval is 192 ms for 6 samples. Use this frame timing to poll the plethysmogram samples.

Note: The Pleth curve-data (PoxPleth) are streamed permanently. The data becomes invalid when AppStatus ≠ measuring. Therefore, Bit 3 is linked to AppStatus and has to be evaluated if the Pleth data shall be saved / displayed (see section 5.2).

5.3 Alarm Handling

Each requested measurement value will also transport alarm status. There is each a bit indicating high or low alarm limit violation condition. The alarm limits as set on the SDM are applied to indicate an alarm condition.

Vital data has either high or medium priority: SpO2 alarm limit violation will trigger a high priority alarm and PCO2, PO2 and PR alarm limit violation will trigger a medium priority alarm.

All technical alarms are low priority alarms or rated as information, with the exception of battery critical alarm that is a medium priority alarm.

ID	Name	Description	Type / Size	Access	Unit / Coding / Range / Example / Default	Change / Data rate
0x87	**AppAlarm**	Reflects status of highest actual alarm level	Enum 1 byte	R-,R-,RW	AppAlarm="info" 0: "info" 1: "low" alarm 2: "medium" alarm	Typically 1 sec (On change of AppStatusMsg)

ID	Name	Description	Type / Size	Access	Unit / Coding / Range / Example / Default	Change / Data rate
					3: "high" alarm	
0xB4	**Pco2HighLimit**	Selects the upper alarm limit for pCO2 measurements	Int 2 bytes	RW, RW, RW	0...200 [mmHg] Pco2HighLimit=45 > Pco2LowLimit + 5 [mmHg]	By menu
0xB5	**Pco2LowLimit**	Selects the lower alarm limit for pCO2 measurements	Int 4 bytes	RW, RW, RW	0...200 [mmHg] Pco2LowLimit=30 < Pco2HighLimit - 5 [mmHg]	By menu
0x9F	**Po2HighLimit**	Selects the upper alarm limit for pO2 measurements	Int 2 bytes	RW, RW, RW	0...800 [mmHg] Po2HighLimit=95 > Po2LowLimit + 5 [mmHg]	By menu
0xA0	**Po2LowLimit**	Selects the lower alarm limit for pO2 measurements	Int 4 bytes	RW, RW, RW	0...800 [mmHg] Po2LowLimit=60 < Po2HighLimit - 5 [mmHg]	By menu
0xA2	**PoxSpO2HighLimit**	Selects the upper alarm limit for SpO2 measurements	Int 2 bytes	RW, RW, RW	5...100 [%] PoxSpO2HighLimit=100 > PoxSpO2LowLimit + 5 [%]	By menu
0xA3	**PoxSpO2LowLimit**	Selects the lower alarm limit for SpO2 measurements	Int 4 bytes	RW, RW ,RW	0...95 [%] PoxSpO2LowLimit=85 < PoxSpO2HighLimit - 5 [%]	By menu
0xA5	**PoxPRHighLimit**	Selects the upper alarm limit for PR measurements	Int 2 bytes	RW, RW ,RW	40...250 [bpm] PoxPRHighLimit=140 > PoxPRLowLimit + 10 [bpm]	By menu
0xA6	**PoxPRLowLimit**	Selects the lower alarm limit for PR measurements	Int 4 bytes	RW, RW ,RW	30...240 [bpm] PoxPRLowLimit=50 < PoxPRHighLimit - 10 [bpm]	By menu

5.4 PCO2 Calibration Line and Residual Drift Correction

The residual drift of a PCO2 measurement can be compensated with calibration data after a measurement for display or for further analysis. Especially for longer measurements, it can be beneficial to increase the accuracy of a pCO2 measurement using residual drift correction.

ID	Name	Description	Type / Size	Access / Persistence	Unit / Coding / Range / Example / Default	Change / Data rate
0x47	**Pco2CalibrationLine**	Shows the results of the last calibration Calibration status string Reflects the results of the last calibration. This information can be logged in monitor for sensor / membrane status reporting	String 1..127 bytes	R-, R-, R-	Pco2CalibrationLine =0x0,NaN,0,0,0,0,0,0,3 field 1: cal Time [MPB RTC sec] (0xhhhhhhhh) field 2: cal Drift [mmHg/h] (xxx.xx) field 3:reserved field 4: cal Interval [h] (xxx.xx) field 5: cal Barometer [mmHg] (xxx.x) field 6: cal Temperature [°C] (xxx.x) field 7: reserved field 8: cal Duration [min] (xx.xx)	After Calibration or Leak Test

ID	Name	Description	Type / Size	Access / Persistence	Unit / Coding / Range / Example / Default	Change / Data rate
					field 9-11: reserved field 12: Validity 0 = ok 1 = leak test not terminated 2 = leak 3 = error (see status code) 4 = leak test will follow	
0x46	**Pco2Progress**	Shows the estimated and the already used calibration time, as well as status of different calibration activities	Int 2+1 bytes	R-,R-,RW	Format: hexadecimal Pco2Progress=0|40 Bit 31..16: estimated duration time Bit 15...0: counted-up time since start Attribute: bit 7: hidden process is active bit 6: manual calibration possible bit 5: sensitivity check active bit 4: Slope check active bit 3: extended calibration (dU/dT) bit 2: gas blast active bit 1: leak test required or active bit 0: calibration required or active	1 sec while calibrating
0x56	**MpbRtc** Should be recorded along with the two calibration lines (please see description below)	Sets and monitors the on-board RTC clock. This parameter must be set at power-up	Int 4 bytes	RW,RW,RW	[RtcSec] MpbRtc=0	At power-up 1 sec

For residual drift correction 3 values of `Pco2CalibrationLine` are needed: field 1 for *calibration time [s]*, field 2 for *drift [mmHg/h]* and field 4 for *calibration interval [h]*. Additionally, the *time stamp* (time base of client) when leak test has terminated (successful) and the corresponding *MPB time* is needed (as requested with `MpbRtc`). A description of the context and relationship of these data points is shown in figure below. Using the data points, linear drift compensation can be achieved. Each data point in time of a previous measurement is compensated by a linear interpolation of the *drift* over the *calibration interval*.

t_2 can be calculated in the time base of the client by using the *time stamp* t_{0_client}, the *MPB time stamp* t_0 and the *calibration time stamp* t_1 (Note: the SDM times are in hexadecimal notation and are represented in seconds: *calibration time* and *MPB time*):

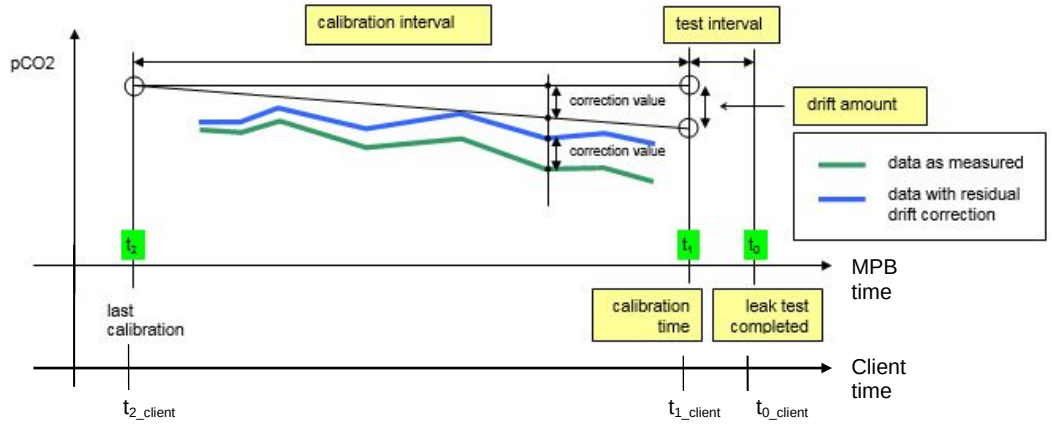
test interval = MPB time at t_0 *– calibration time* t_1

$t_{1\_client} = t_{0\_client}$ – *test interval*
$t_{2\_client} = t_{1\_client}$ – *calibration interval*

All measurement data in the time base of the client in the interval t_{2_client} to t_{1_client} can now be linearly compensated:

data_compensated(t) = data(t) + *drift* * (t – t_{2_client}) ; all data(t) in time base of client

Note: A conversion into the time base of the client (t_{0_client}, t_{1_client} and t_{2_client}) is important, because the SDM clock and client clock will never match exactly and will drift from each other over longer periods.



t_{0_client}, t_{1_client} and t_{2_client}: client time stamp

t_0: MPB time stamp at t_0 [s] (*MpbRtc*)
t_1: *calibration time* stamp in MPB time [s] (field 1)
t_2 : MPB time stamp at last calibration [s]

calibration interval [h] (field 4)
drift amount [mmHg/h] (field 2)

Note: The Sentec Digital Monitor is able to output a "Full Memory Dump" (via command: "/historical_data/get/") where the PCO2 values as well as drift corrected PCO2 values of (a) measurement(s) are shown in columns. A validation of the calculated, drift corrected PCO2 values can be made by doing a comparison between the SDM output and the calculated values.

5.5 Sensor Temperature, Heating Power and related settings

ID	Name	Description	Type / Size	Access	Unit / Coding / Range / Example / Default	Change / Data rate
0x23	WuSetTemp	Selects the probe temperature. The range depends on AppPatientType This parameter may change to a new default value, when the patient type is changed or when a new type of sensor is connected.	Float 4 bytes	RW, RW, RW	37.0…44.5 [°C] WuSetTemp=42.0 Steps: 0.5 [°C]	By menu and AppPatientType
0x24	WuTemp note the quality bits 0 to 7 attributed to this object, in particular bits 4 to 7 for the general quality	Shows the actual sensor temperature The Quality attribute bits [0..3] may be used to select a mode Icon. The general Quality attribute may be used for hiding the value when not available, setting to '---' when invalid or grey shading when unstable or tagging with a '?' when questionable.	Float 4+1 bytes	R-,R-,R-	0.0…120.0 [°C] WuTemp=0.0|40 Quality attribute (in hex format): Bit 0: Temp Limiter is active (OT) Bit 1: Initial Heating is active (IH) Bit 2: Site Protection is active (SP) Bit 3: reserved Bit 4…7: general Quality	Typically 1 sec (WuDisplayPeriod)
0x25	WuPowerMode	menu parameter, Referencing relative power measurement is automated and gets valid, as soon as stable temperature and measurements are available. Toggling WuPowerMode will update the reference value as soon as the measurement conditions will allow it.	Enum 1 byte	RW, RW, RW	WuPowerMode='relative' 0: 'off' 1: 'relative' 2: 'absolute'	By menu
0x2B	WuPowerRef	Shows actual power reference value	Int 2 bytes	R-,R-,R-	0…999 [mW] WuPowerRef=0	In relative mode after 1..5min measuring
0x26	WuPower note the quality bits 0 to 7 attributed to this object, in particular bits 4 to 7 for the general quality to a certain extent an indirect measure of local skin blood flow	Shows the actual absolute heating power Total sensor power consumption depending on Sensor Type, Version and WuPowerMode. In "absolute" mode, the effective power is measured. In "relative" mode, effective power is reduced by the value, measured at reference time-point. See WuPowerMode for selecting the reference time-point. The Quality attribute bits [0…3] may be used to select a mode Icon. The general Quality attribute may be used for hiding the value when not available, setting to '---' when invalid or grey	Int 2+1 bytes	R-,R-,R-	0…999 [mW] WuPower=0|40 Quality attribute (in hex format): Bit 0…1: reserved Bit 2: Power referencing allowed Bit 3: Relative Measurement Bit 4..7: general Quality	WuDisplayPeriod (typically 1 sec)

ID	Name	Description	Type / Size	Access	Unit / Coding / Range / Example / Default	Change / Data rate
		shading when unstable or tagging with a '?' when questionable.				

5.6 Monitor/ Sensor specific information

ID	Name	Description	Type / Size	Access / Persistence	Unit / Coding / Range / Example / Default	Change / Data rate
0xF9	SmbSerialNb	Shows serial number of SDM as stored in the motherboard EEPROM.	Int 4 bytes	R-,R-,RW	SmbSerialNb=0	By SentecLink (or SmbSerialNb)
0xFB	SmbSwRelease	Shows the actual software version as published	String 1..31 bytes	R-,R-,R-	SmbSwRelease="V08.03.x"	Never
0x53	MpbSwRelease	Shows the actual software version as published	String 1..31 bytes	R-,R-,R-	MpbSwRelease="V06.03.xx"	Never
0x64	SensorName	Shows the Type name of the sensor	String 1..31 bytes	R-,R-,R-	SensorName="---" "---" when no sensor is connected "VS-A/P/N" (for VSign2 sensor) "OV-A/P/N" (for OxiVenT™ sensor)	On sensor connect
0x61	SensorSerialNb -	Shows serial number of the connected sensor	Int 4 bytes	R-,R-,R-	SensorSerialNb=0	On sensor connect
0x63	SensorSwRelease	Shows the actual software version as received from connected Sensor	String 1..31 bytes	R-,R-,R-	SensorSwRelease="-"	On sensor connect

5.7 Multiple LAN Clients

If there is a possibility that more than one client will be accessing the data from an SDM via the network (LAN), this must be managed carefully. The SDM can only process a limited number of UDP request per time unit. If this amount is exceeded, additional frames will be discarded unprocessed. In order to manage

the access to an SDM there is a token-based cooperative locking mechanism (see example in section 6.5). The table below shows the necessary data objects used to manage data traffic load and token based locking.

Use `LanRejectionRate` and `LanProcessingRate` mainly in the development phase of a client interface. Try to keep `LanProcessingRate` below 500 frames. For productive use of the interface the monitoring of `LanRejectionRate` will help to detect traffic or configuration problems in the network, e.g. a high broadcast load will count towards the processed frames and might trigger the rejection of frames and therefore affect the reliability of the established connection. `LanRejectionRate` should normally remain at zero.

ID	Name	Description	Type / Size	Access	Unit / Coding / Range / Example / Default	Change / Data rate
-	LanRejectionRate	This is the number of Ethernet frames that are rejected within a time interval of 10sec.	Int 2 bytes	R-,R-,RW	LanRejectionRate=0	10sec
-	LanProcessingRate	This is the number of Ethernet frames that are processed within a time interval of 10sec.	Int 2 bytes	R-,R-,RW	LanProcessingRate=0	10sec
0x98	AppRemoteTimeout	Configures the timeout for periodic RemoteLock update	Int 2 bytes	RW, RW, RW	[sec] AppRemoteTimeout=5	By remote application
0x99	AppRemoteLock	Should be periodically written by a remote application for exclusively locking the measurement device for remote monitoring. The lock is reset by the SDM after AppRemoteTimeout. A remote application has to refresh the lock periodically by writing a unique number. If the device is already locked by another remote application, the number of the locking application is returned	Int 4 bytes	RW, RW, RW	AppRemoteLock=0 >0: Write any positive unique number for locking the device. Reading back the very same lock number acknowledges a previously granted lock for this remote application. Reading back another lock-number means, the device is already locked by another remote application. A measurement session then should not be opened. <0: Reading back the negated lock number indicates that, the device was newly locked for this remote application. This answer can be used to set AppRemoteRef. 0: Writing 0 unlocks the device	By remote application
0x9A	AppRemoteRef	Holds reference information about the patient or the measurement application. This parameter is used in companion with a remote measurement application and is shown on both, the device's screen and the remote screen.	String 1..127 bytes	RW, RW, RW	AppRemoteRef=""	By remote application

ID	Name	Description	Type / Size	Access	Unit / Coding / Range / Example / Default	Change / Data rate
		The remote application is free to set any string value after successfully locking the device. As soon as AppRemoteLock = 0, this string is cleared by the SDM.				

5.8 Lan Device Discovery

Requesting for Devices, Device Discovery

Sentec SDM Devices on the network can be detected by sending a broadcast UDP packet to port 68 (BOOTP/DHCP) with the binary content as described below (similar to DHCP discovery frames, with Sentec specific magic cookie):

Byte [0]	0x02
Byte [1]	0x01
Byte [2]	0x06
...	
Byte [236]	0x53 'S'
Byte [237]	0x44 'D'
Byte [238]	0x4D 'M'
Byte [239]	0x53 'S'
Byte [240]	0x01
Byte [241]	0x01
...	
Byte [244]	MSB of client port number
Byte [245]	LSB of client port number
Byte [246]	ScanMask byte [0]
Byte [247]	ScanMask byte [1]
...	...
Byte [246+n]	ScanMask byte [n]
Byte [246+n+1]	0x00

All other bytes of the frame have to be filled with zeros. The frame may be length padded with zeros.

In order to scan for specific devices, a null-terminated ScanMask string can be sent with the broadcast. Only those devices with a complete match of the ScanMask string with the beginning of the device's LanHostname will answer the request.

Sentec_SDM64

this scan string example will report all SDM devices with a serial number beginning with 64xxxx.

Scanned Device Answer

Reached devices with a matching `LanHostname` will answer to the broadcasting client's specified UDP port address. The following string Command will be sent with the answer:

DeviceName=nnn.nnn.nnn.nnn:pppp

Nnn.nnn.nnn.nnn is the IP address of the device and pppp is the actually configured SMI protocol port.

6. Examples on how to communicate with the SDM

The following examples explain the different states during communication by using SMI commands. Therefore, two state machines are defined – state machine one with phase 0 – 4 that explains how to establish communication and request vital data and the second state machine consists of phase A – D that describes a lock mechanism for the LAN interface. Phases 0 – 4 of the first state machine describe following states:
0 – initialization requirement
1 – ongoing communication when waiting for measurement to start
2 – ongoing communication during measurement
3 – ongoing communication after measurement
4 – requesting calibration data after measurement

Phases A – D of state machine 2 describes following states:
A – check lock status
B – lock
C – maintain / refresh lock
D – release lock

6.1. Example – Starting a measurement and requesting primary parameters

Phase 0: *Initialization requirement:*

Send:	`Pco2DisplayPeriod`	; request PCO2 channel data timing
Response:	`Pco2DisplayPeriod=960`	; example with 960 ms PCO2 data timing (default)
Send:	`PoxDisplayPeriod`	; request POX channel data timing
Response:	`PoxDisplayPeriod=992`	; example with 992 ms POX display timing (default)

As seen from above default values for data timing, new data is available approximately every second and should be polled accordingly. Over sampling will return the same value again.

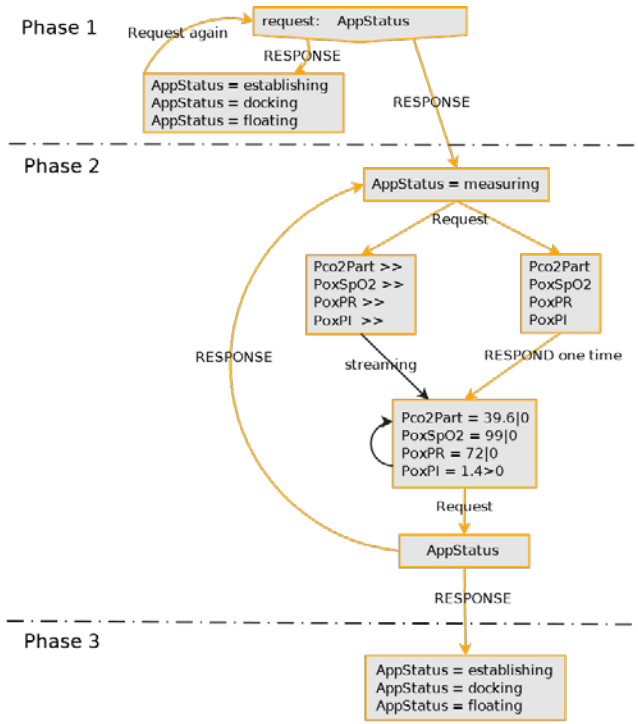
Phase 1*: Ongoing communication when waiting for a measurement to start:*

Send:	`AppStatus`	; request status of sensor
Response:	`AppStatus=establishing`	; no sensor attached, no measurement possible (quality bits for all channels set to 4 (not available))
Response:	`AppStatus=docking`	; sensor in docking station, no measurement active (quality bits for all channels set to 3 (invalid))
Response:	`AppStatus=floating`	; sensor has been removed from docking station (quality bits for all channels set to 3 (invalid))
Response:	`AppStatus=measuring`	; sensor on patient and measuring (quality bits see examples below), transition to Phase 2 Possible

As soon as `AppStatus=measuring` is reported *Phase 1* is terminated and *Phase 2* becomes active.

Phase 2: *Ongoing communication during a measurement:*

Send:	`AppStatus`	; request status of sensor
Response:	`AppStatus=measuring`	; check whether measurement is still active
Send:	`Pco2Part`	; request PCO2 value (arterial PCO2)
Response:	`Pco2Part=39.6|0`	; PCO2 data and quality (see section 4.3 for meaning of quality bits)
Send:	`PoxSpO2`	; request SpO2 data
Response:	`PoxSpO2=99|0`	; SpO2 data and quality (see section 4.3 for meaning of quality bits)
Send:	`PoxPR`	; request pulse rate data
Response:	`PoxPR=72|0`	; pulse rate data and quality (see section 4.3 for meaning of quality bits)
Send:	`PoxPI`	; request pulsation index data
Response:	`PoxPI=1.4|0`	; pulsation index data and quality (see section 4.3 for meaning of quality bits)



Phase 3: *Ongoing communication after measurement:*

Send: `AppStatus` ; request status of sensor
Response: `AppStatus=floating` ; measurement is inactive

To speed up communication on LAN interface the commands can also be grouped as recommended for UPD frames. To collect vital data a frame could be composed as follows:

```
vital_data_frame = "Pco2Part<CR><VT><CRC>PoxSpO2<CR><VT><CRC>PoxPR<CR><VT><CRC>PoxPI<CR><VT><CRC>"
```

The SDM will respond in sequence.

If anything else than `AppStatus=measuring` is reported to `AppStatus` go to Phase 3 which is in this case equal to *Phase 1*, waiting for a new measurement to start.

6.2. Example – requesting pleth values

Phase 1: *Ongoing communication when waiting for a measurement to start:*

Same as for Vital Data, see above.

Phase 2: *Ongoing communication during a measurement:*

Send: `PoxPleth` ; request pleth data
Response: `PoxPleth=6,0f28,0ebb,0e45,0dd1,0d7c,0d85` ; 6 samples with flags as described in section 5.2

With a frame timing of 192 ms and the underlying sample spacing of 32 ms 6 pleth values have to be processed per frame. Make sure that for buffering and plotting routines this fixed timing rate is followed. **In case of over sampling, make sure to drop a duplicate frame to avoid duplicate plotting.**

Verify bit 14 for validity of the sample value. Do not plot invalid data samples or provide zero line instead. In addition, if pulse beep is required bit 15 can be used as a trigger for generating an audible pulse beep.

Note: Timing for plethysmogram data is different from vital data; make sure to run Phase 2 for ongoing communication with the specified timing.

Note: When running a client connection via UDP, then note that network load or latency can affect the performance of data transfer. Make sure to buffer this data point and provide means to detect data underrun.

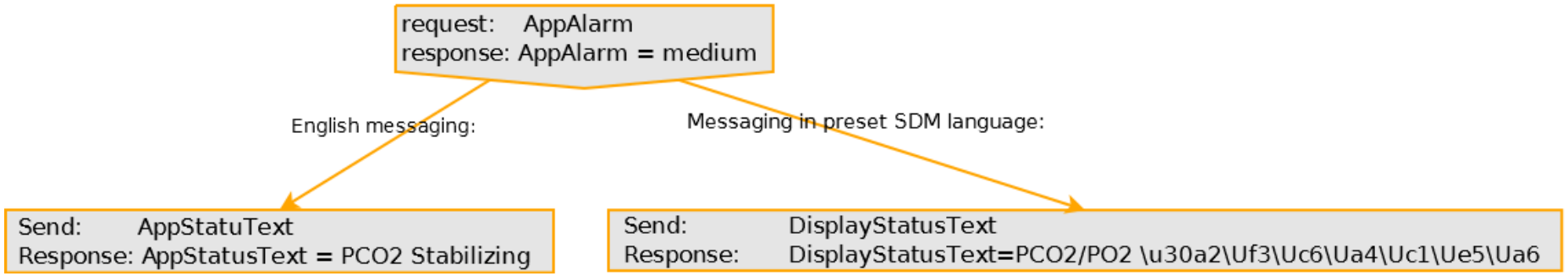
6.3. Example – requesting alarms

Phase >0: Ongoing communication during a measurement:

Send: `AppAlarm` ; request alarm status
Response: `AppAlarm=medium` ; status info message → check quality bits of vital data for source of alarm

English messaging:
Send: `AppStatusText` ; request message
Response: `AppStatusText=pCO2/po2 Stabilizing` ; status info message
Messaging in preset SDM language:
Send: `DisplayStatusText` ; request message
Response: `DisplayStatusText= PCO2/PO2 \u30a2\Uf3\Uc6\Ua4\Uc1\Ue5\Ua6` ; status info message in Unicode Japanese

Note: "DisplayStatusText" messages are coded in Unicode that follows special rules. A Unicode character starts with a "\u" followed by 2 Bytes or with "\U" followed by the LSB. Using the MSB from the last used "\u" character.



6.4. Example – requesting calibration data

Calibration data is available for measurements lasting for more than 10 minutes. After a transition from *Phase 2* (measuring) to *Phase 3* (waiting for a measurement) the following command sequence applies.

Phase 2: *Ongoing communication during measurement:*

Send: `AppStatus` ; request status of sensor
Possible responses:
Response: `AppStatus=establishing` ; sensor has been disconnected (quality for all channels set to 4 (not available))
Response: `AppStatus=floating` ; sensor has been removed from patient (quality for all channels set to 3 (invalid))
Response: `AppStatus=measuring` ; sensor on patient and measuring (quality bits see section 4.3), transition to *Phase 2* possible

Phase 3: *Ongoing communication after a measurement:*

After a measurement, the responses will normally go through this sequence:
Response: `AppStatus=floating` ; sensor has been removed from patient (quality for all channels set to 3 (invalid))
Response: `AppStatus=docking` ; sensor has been put in docking station

As soon as `AppStatus=docking` is reported after a measurement lasting for more than 10 minutes a sensor calibration is started. This can be verified using this command with

Send: `Pco2Progress` ; request calibration status
Possible responses:
Response: `Pco2Progress=0x00870006|1` ; response after 6 seconds of calibration, note that attribute bit 0 is one (= calibration active)

Bit 0 of attribute byte indicates active calibration. As soon as bit 0 resets, calibration has terminated and a leak test is performed. Bit 1 of attribute byte indicates a leak test that is performed after each calibration. A complete calibration cycle has completed once the leak test has terminated (indicated with the reset of bit 1).

Request this object every second until completion of calibration (bit 0 resets to 0). If a full calibration cycle including leak test is required then continue to poll this value until bit 1 resets to 0, otherwise skip this next step and directly proceed to *Phase 2*

Response: `Pco2Progress=0x00000000|c2` ; response after calibration, leak test running

Termination of calibration and leak test is indicated as follows:
Response: `Pco2Progress=0x00000000|0` ; response after calibration and leak test
Note that other bits can also have a value in the attribute byte:
Response: `Pco2Progress=0x00000000|10` ; response after calibration and leak test (slope check active)
Response: `Pco2Progress=0x00000000|50` ; response after calibration and leak test (slope check and manual calibration possible)

Note: Instead of all zero values in the left part of the value returned can be non-zero numbers (see section Q), where the first 16 bits are an estimate of the time needed for calibration while the last 16 bits are an up counter of the actual used time.

Phase 4: *Requesting calibration data:*
Once calibration (and optionally leak test) have terminated, the calibration data needed for residual drift correction can be retrieved as follows (request information immediately after Pco2Progress indicates completion):
Send: `Pco2CalibrationLine` ; request calibration line
Response: `Pco2CalibrationLine=0x1499ff63,-0.17,0.47,0.60,737.3,41.9,-0.347,3.33,169.2,0.80,0` ; calibration data
Send: `MpbRtc` ; request MPB time (time of Multi Parameter Board)
Response: `MpbRtc=149A00CB` ; MPB time (after successful or aborted leak test)

First check last field of the `Pco2CalibrationLine` response which indicates the status of the calibration: if value = 0 then calibration terminated without an error, calibration data is valid. *Please note:* different versions of the software may have a different number of data fields reported by `Pco2CalibrationLine`, the status will always be the last field. If status value is 1 then leak test has been interrupted, e.g. the user has opened the docking station door to remove the sensor. If status value is 4 then leak test is currently running. You can use the calibration data for residual drift correction in these three cases (status = 0 | 1 | 4), see section 5.4.

6.5. Example – Acquiring and maintaining a lock (token):

Phase A: Check Lock status:

Send:	`AppRemoteLock`	; request lock status
Response:	`AppRemoteLock=0`	; status of lock (unlocked in this case)

If zero is returned, continue with *Phase 1*, otherwise device is already locked by another client.

Phase B: Acquire Lock:

Send:	`AppRemoteLock=<32-bit integer random number>`	; set lock
Response 1:	`AppRemoteLock=-<32-bit integer random number>`	; confirm lock status
Response 2:	`AppRemoteLock=<another 32-bit integer random number>`	; lock status not confirmed

If the response is the same random number with negative sign, then the lock is confirmed, continue with *Phase 2*. If the number is any other integer, then another client acquired the token prior to this request, back to *Phase 0*.

Phase C: Maintain / Refresh Lock:

Send:	`AppRemoteLock=<32-bit integer random number>`	; refresh lock
Response 1:	`AppRemoteLock=<32-bit integer random number>`	; lock status refreshed / confirmed
Response 2:	`AppRemoteLock=-<32-bit integer random number>`	; lock lost, refresh not sent within lock timeout period, reacquired lock
Response 3:	`AppRemoteLock=<another 32-bit integer random number>`	; lock lost, refresh not sent within lock timeout period, another client now owns the token

The refresh of the lock has to take place within `AppRemoteTimeout` otherwise the lock condition will be reset and another client can access the lock (response 3). In the case, that the lock was lost and the response includes a negative sign then the lock was reacquired (response 2). `AppRemoteTimeout` can be set as needed (e.g. depending on traffic latencies).

Phase D: Release Lock:

Send:	`AppRemoteLock=0`	; release lock
Response:	`AppRemoteLock=0`	; lock release confirmed

Setting customized info string on SDM:

During *Phase B* right after lock was confirmed, remote client could set the customized info string. This string will be displayed on the SDM (see technical manual for all occurrences).

Send: `AppRemoteRef=Ward 123 ; set string`
Response: `AppRemoteRef=Ward 123 ; confirmed`