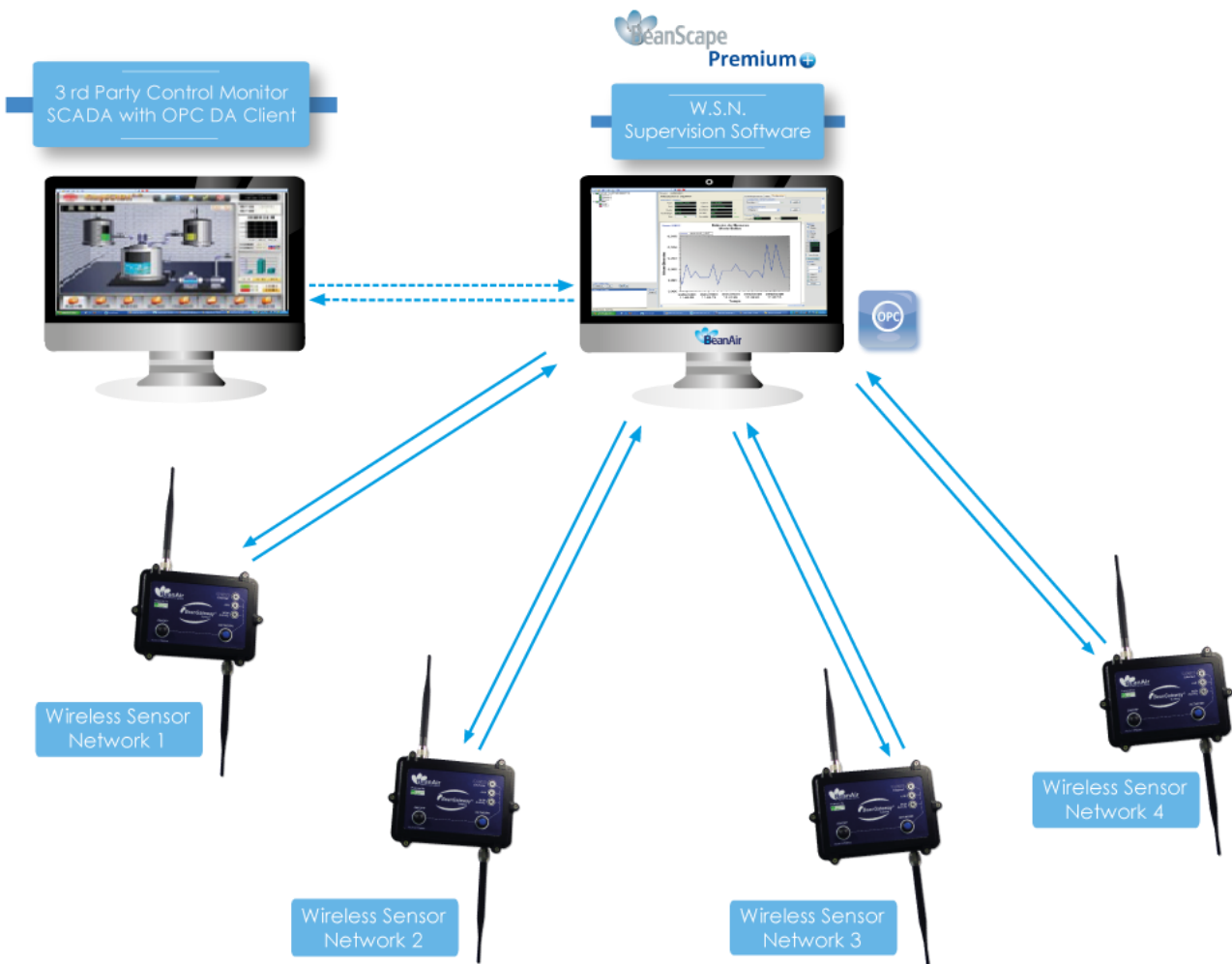




Version 1.6.1

USER MANUAL

OPC Server add-on (BeanScope® Premium+)



DOCUMENT

Document number		Version	V1.6.1
External Reference	UM_RF_008	Publication date	09/05/2019
Author	Fahd ESSID, Embedded Software Engineer		
Internal Reference		Project Code	
Document Name	OPC Server		

VALIDATION

Function	Recipients	For Validation	For information
Reader	Maxime Obraztsov		X
Author	Maneli PARSY	X	

MAILING LIST

Function	Recipients	For action	For Info
Staffer 1	Maneli PARSY	X	
Staffer 2	Christophe DONTGREUIL		X

Updates

Version	Date	Author	Evolution & Status
1.3.3	06/07/2012	Maneli PARSY	<ul style="list-style-type: none"> • "VT_BEAN_PLATFORM_TYPE" added
1.3.5	28/11/2014	Mohamed Yosri Jaouadi	<ul style="list-style-type: none"> • Burst mode, Oneshot mode and Commissioning Mode command inserted
1.4	20/01/2015	Mohamed Yosri Jaouadi	<ul style="list-style-type: none"> • Datalogger full memory strategy commands inserted
1.5	10/01/2016	Mohamed Yosri Jaouadi	<ul style="list-style-type: none"> • « VT_Acknowledgement » added • « VT_DownloadStatus » added • « VT_LoggerStatus » added
1.6	20/12/2018	Fahd Essid	<ul style="list-style-type: none"> • Chart update • Screenshots update • Vocabulary update • OTAC commands update • Commissioning timeout command deleted • OPC DA Client source code added
1.6.1	09/05/2019	Mohamed Bechir Besbes	<ul style="list-style-type: none"> • Weblinks Update



Contents

1.	TECHNICAL SUPPORT	7
2.	VISUAL SYMBOLS DEFINITION	8
3.	ACRONYMS AND ABBREVIATIONS	9
4.	RELATED DOCUMENTS	10
4.1	Application Notes	10
4.2	Technical Notes	11
5.	GENERAL OVERVIEW	12
5.1	BeanScape Premium+ Block Diagram	12
5.2	Service OPC DA (Data Access).....	12
6.	CONFIGURATION AND LAUNCHING OF THE OPC SERVER FROM THE GUI.....	14
6.1	OPC TAB ACCESS.....	14
6.2	OPC Parameters.....	15
6.2.1	General Configuration	15
6.2.2	Streaming Configuration	16
6.3	Start and Stop the OPC Server.....	19
7.	DETAILED DATA DESCRIPTION	20
7.1	Data Tree Presentation: OPC DA XPAC.....	20
7.2	Data Tree Presentation: OPC DA XTEND	22
7.3	Data Related to the Beandevice	24
7.3.1	“BeanDevice®” Item/Tag.....	24
7.3.2	Sleep Mode/Power Mode attribute OPC DA XPAC.....	24
7.3.3	Sleep Mode/Power Mode Tag/Item OPC DA XTEND	24
7.3.4	Diagnostic Attributes: OPC DA XPAC	25
7.3.5	The Platform Type Attribute OPC D	26
7.3.6	A XPAC	26
7.3.7	The Platform Type Tag/Item OPC DA XTEND.....	26
7.3.8	The acknowledgement Tag/Item	27
7.3.9	DownloadStatus Item/Tag.....	27
7.3.10	LoggerStatus Item/Tag.....	28
7.3.11	Diagnostic Items/Tags: OPC DA XTEND.....	28
7.3.12	The « Command » Item/Tag	29
7.4	Data Related to the Sensor	30

7.4.1	“STREAMING” Item/Tag	30
7.4.2	“BEANSENSOR” Item/Tag.....	31
7.4.3	« Type » attribute: OPC DA XPAC	31
7.4.4	« Type » Item/Tag: OPC DA XTEND.....	32
8.	DETAILED DESCRIPTION OF BENGATEWAY COMMANDS.....	33
8.1	System Acknowledgment	33
8.2	Commands for Configuring the Gateway	33
8.2.1	The “SetClockBroadcastCycle” Command	33
8.2.2	The “SetNetwkDiagCycle” Command	34
8.2.3	The “SetTxPowerConfig” Command.....	35
8.2.4	The “EraseGatewayNetworkContext” Command	36
9.	DETAILED DESCRIPTION OF BEANDEVICE COMMANDS.....	38
9.1	System Acknowledgment	38
9.2	Commands for setting the measure mode	38
9.2.1	The “SetMeasureMode_Survey” Command	38
9.2.2	The “SetMeasureMode_Survey_WLogger” Command	39
9.2.3	The “SetMeasureMode_LowDutyCycle” Command	40
9.2.4	The “SetMeasureMode_LowDutyCycle_WLogger” Command	41
9.2.5	The “SetMeasureMode_StreamingPacket” Command.....	42
9.2.6	The “SetMeasureMode_StreamingPacket_CntMon” Command.....	43
9.2.7	The “SetMeasureMode_Streaming_CntMon” Command	44
9.2.8	The “SetMeasureMode_SSD” Command	45
9.2.9	The “SetMeasureMode_SSD_CntMon” Command.....	46
9.2.10	The “SetMeasureMode_StreamingPacket_Burst” Command.....	47
9.2.11	The “SetMeasureMode_StreamingPacket_Burst” Command.....	48
9.2.12	The “SetMeasureMode_StreamingPacket_OneShot” Command	50
9.2.13	The “SetMeasureMode_StreamingPacket_OneShot” Command	51
9.2.14	The “SetMeasureMode_Commissioning” Command.....	52
9.3	Commands for Configuring the Dignostics and TX Power Mode	53
9.3.1	The “SetNetwkDiagCycle” command.....	53
9.3.2	The “SetTxPowerConfig” command.....	53
9.4	Leave Network command.....	54
9.5	Commands for Configuring the Sleep Mode	55
9.5.1	The “SetWaitingFrameDeletion” command.....	55
9.5.2	The “EnableSleepMode” command	56
9.5.3	The “DisableSleepMode” command	56
9.5.4	The “EnableSleepWithListening” command.....	57
9.6	Commands for configuring the Data Logger.....	58
9.6.1	“Logger_Download_Data” command.....	58
9.6.2	“Logger_Stop_Logging_Data” command.....	59
9.6.3	“Logger_Erase_Stored_Log” command.....	59
9.6.4	“Logger_Cancel_Upload” command.....	60
9.6.5	DownloadThenErase Command	61
9.6.6	SCDE Command	61
9.7	Data logger full memory strategy COMMANDS:	62
9.7.1	“SetFullMemory_SC” command :	62

9.7.2	“SetFullMemory_SAE” command :	63
9.7.3	“SetFullMemory_SCDE” command :	64
10.	DETAILED DESCRIPTION OF THE TAG VALUE	65
11.	DETAILED DESCRIPTION OF STANDARD TAG ATTRIBUTES “DATE” AND “QUALITY”	66
11.1	Date Attribute	66
11.2	Quality Attribute	66
11.2.1	Sensor Quality Determination	68
11.2.2	BeanDevice® Quality Determination	69
11.2.3	BeanGateway® Quality Determination	70
11.2.4	BeanScape® Quality Determination	70
11.2.5	Command tag quality determination	70
12.	APPENDICES	71
12.1	Appendice1: First steps to develop your OPC Client with C#	71

Disclaimer

The information contained in this document is the proprietary information of Beanair.

The contents are confidential and any disclosure to persons other than the officers, employees, agents or subcontractors of the owner or licensee of this document, without the prior written consent of Beanair GmbH, is strictly prohibited.

Beanair makes every effort to ensure the quality of the information it makes available. Notwithstanding the foregoing, Beanair does not make any warranty as to the information contained herein, and does not accept any liability for any injury, loss or damage of any kind incurred by use of or reliance upon the information.

Beanair disclaims any and all responsibility for the application of the devices characterized in this document, and notes that the application of the device must comply with the safety standards of the applicable country, and where applicable, with the relevant wiring rules.

Beanair reserves the right to make modifications, additions and deletions to this document due to typographical errors, inaccurate information, or improvements to programs and/or equipment at any time and without notice.

Such changes will, nevertheless be incorporated into new editions of this document.

Copyright: Transmittal, reproduction, dissemination and/or editing of this document as well as utilization of its contents and communication thereof to others without express authorization are prohibited. Offenders will be held liable for payment of damages. All rights are reserved.

Copyright © Beanair GmbH 2016

1. TECHNICAL SUPPORT

For general contact, technical support, to report documentation errors and to order manuals, contact **Beanair Technical Support Center** (BTSC) at:
tech-support@Beanair.com




For detailed information about where you can buy the Beanair equipment/software or for recommendations on accessories and components visit:

www.Beanair.com

To register for product news and announcements or for product questions contact Beanair's Technical Support Center (BTSC).

Our aim is to make this user manual as helpful as possible. Please keep us informed of your comments and suggestions for improvements. Beanair appreciates feedback from the users.

2. VISUAL SYMBOLS DEFINITION

<i>Visual</i>	<i>Definition</i>
	<u>Caution or Warning</u> – Alerts the user with important information about Beanair wireless sensor networks (WSN), if this information is not followed, the equipment /software may fail or malfunction.
	<u>Danger</u> – This information MUST be followed if not you may damage the equipment permanently or bodily injury may occur.
	<u>Tip or Information</u> – Provides advice and suggestions that may be useful when installing Beanair Wireless Sensor Networks.

3. ACRONYMS AND ABBREVIATIONS

AES	Advanced Encryption Standard
CCA	Clear Channel Assessment
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
GTS	Guaranteed Time-Slot
kSps	Kilo samples per second
LLC	Logical Link Control
LQI	Link quality indicator
LDCDA	Low duty cycle data acquisition
MAC	Media Access Control
PAN	Personal Area Network
PER	Packet error rate
RF	Radio Frequency
SD	Secure Digital
SSD	Smart shock detection
WSN	Wireless sensor Network

4. RELATED DOCUMENTS

In addition to this *User Manual*, please consult the application notes & technical notes mentioned below:

4.1 APPLICATION NOTES

Document name (Click on the weblink)	Related product	Description
<u>AN_RF_007 :“ Beanair WSN Deployment”</u>	All BeanAir products	Wireless sensor networks deployment guidelines
<u>AN_RF_006 – „How to extend your wireless range“</u>	All BeanAir products	A guideline very useful for extending your wireless range
<u>AN_RF_005 – BeanGateway® & Data Terminal Equipment Interface</u>	BeanGateway®	DTE interface Architecture on the BeanGateway®
<u>AN_RF_003 - “IEEE 802.15.4 2.4 GHz Vs 868 MHz”</u>	All BeanAir products	Comparison between 868 MHz frequency band and a 2.4 GHz frequency band.
<u>AN_RF_002 – “Structural Health monitoring on bridges”</u>	All BeanAir products	The aim of this document is to overview Beanair® products suited for bridge monitoring, their deployment, as well as their capacity and limits by overviewing various Data acquisition modes available on each BeanDevice®.

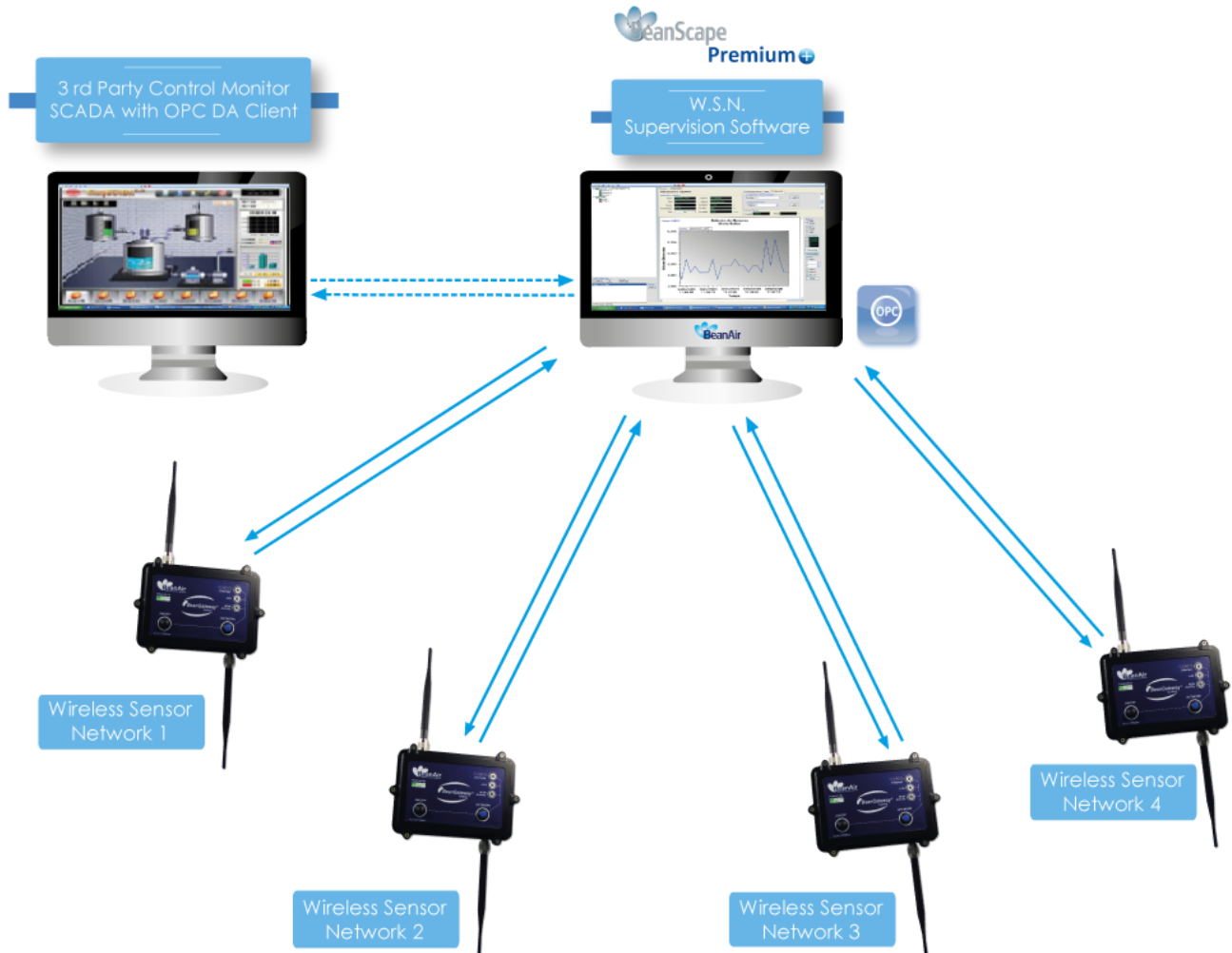
4.2 TECHNICAL NOTES

Document name (Click on the weblink)	Related product	Description
<u>TN RF 013 – « OPC configuration »</u>	BeanScope® Premium+	The aim of this document is to help deploying the OPC DA and all associated services.
<u>TN RF 012– « BeanDevice® battery life in streaming mode »</u>	All the products	The aim of this document is to describe the autonomy performance of the BeanDevice® SmartSensor® and ProcessSensor® product line in streaming packet mode.
<u>TN RF 011 – « Coexistence of Beanair WSN at 2.4GHz »</u>	All the products	This document aims to highlight the issues affecting co-existence of Beanair WSN (IEEE 802.15.4) in the presence of interference.
<u>TN RF 010 – « BeanDevice® Power Management »</u>	All the BeanDevice®	This technical note describes the sleeping & active power mode on the BeanDevice®.
<u>TN RF 009 – « BeanGateway® management on LAN infrastructure »</u>	BeanGateway®	BeanGateway® integration on a LAN infrastructure
<u>TN RF 008 – “Data acquisition modes available on the BeanDevice®”</u>	All the BeanDevice®	Data acquisition modes available on the BeanDevice®
<u>TN RF 007 – “BeanDevice® DataLogger User Guide ”</u>	All the BeanDevice®	This document presents the DataLogger feature on the BeanDevice®
<u>TN RF 006 – “WSN Association process”</u>	All the BeanDevice®	Description of the BeanDevice® network association
<u>TN RF 005 – “Pulse counter & binary Data acquisition on the BeanDevice® SUN-BN”</u>	BeanDevice® SUN-BN	This document presents Pulse counter (ex: energy metering application) and binary Data acquisition features on the BeanDevice® SUN-BN.
<u>RF TN 003- “Aggregation capacity of wireless sensor networks”</u>	All the products	Network capacity characterization of Beanair Wireless Sensor Networks
<u>RF TN 002 V1.0 - Current consumption in active & sleeping mode</u>	BeanDevice®	Current consumption estimation of the BeanDevice in active and sleeping mode
<u>RF TN 001 V1.0- Wireless range benchmarking</u>	BeanDevice®	Wireless range benchmarking of the BeanDevice®

5. GENERAL OVERVIEW

5.1 BEANSCAPE PREMIUM+ BLOCK DIAGRAM

The BeanScape® Premium+ OPC DA, integrates an OPC DA server (Data Access).



5.2 SERVICE OPC DA (DATA ACCESS)

The OPC DA (Data Access) is particularly well suited for real time measurement and data sharing. Each data/measurement can be associated to a tag or its attributes and shared with one or many OPC clients. Hence the **BeanScape® Premium+** opens up to many third party applications (SCADA, web portals etc ...).

Note that the **BeanScape® Premium+** OPC is DA 2, DA 2.5 and DA 3 compliant.



Before installing the BeanScope® Premium+ make sure that you have already downloaded and installed « OPC Core Components » (redistributable). You can download it from <https://www.opcfoundation.org>.

Also read the BeanScope® manual carefully before going through the OPC manual.

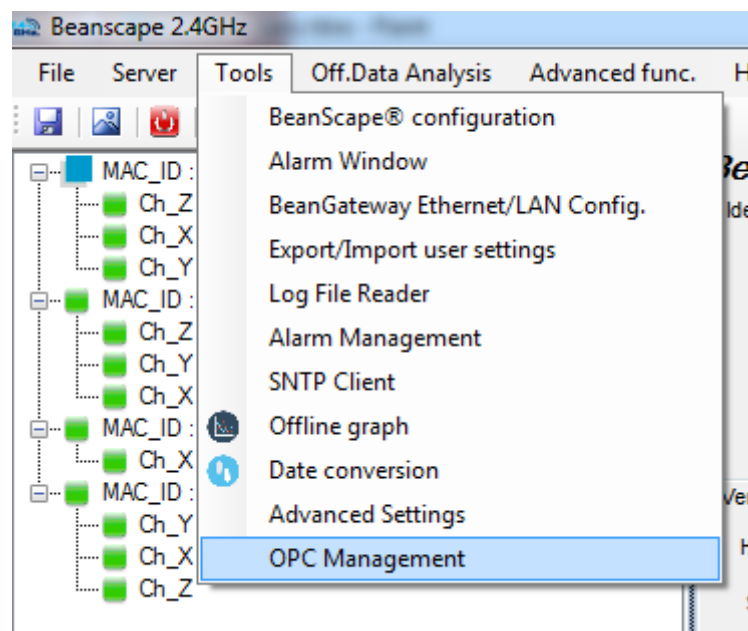
The OPC is available under the following operating modes:

<i>Data Acquisition mode</i>	<i>Data Transmission through OPC DA Channel</i>
Marker	<i>Yes</i>
LowDutyCycle	<i>Yes</i>
Survey	<i>Yes</i>
Streaming Packet	<i>Yes (new feature)</i>
SSD	<i>Yes (new feature)</i>

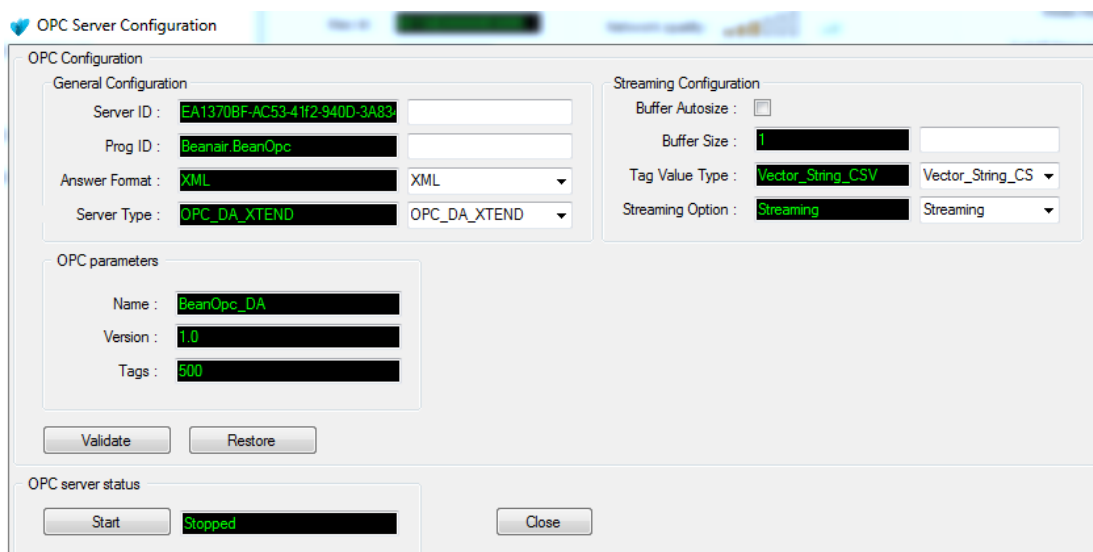
6. CONFIGURATION AND LAUNCHING OF THE OPC SERVER FROM THE GUI

6.1 OPC TAB ACCESS

To start the OPC Server on BeanScape®, you should go to **Tool** on the main menu bar and select **OPC Management**



The OPC Server Configuration window will be displayed.



6.2 OPC PARAMETERS

OPC parameters

Name :

Version :

Tags :

On the OPC Server Configuration you will see the OPC parameters related to your server:

- **Name** : OPC DA name
- **Version** : Indicates the OPC DA version
- **Tags** : Indicates the maximum number of tags available on the OPC DA server.

6.2.1 General Configuration

OPC Configuration

General Configuration

Server ID :

Prog ID :

Answer Format : XML

Server Type : OPC_DA_XTEND

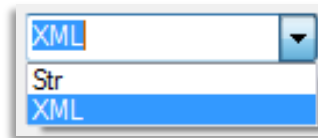
Server ID Field

Prog ID Field

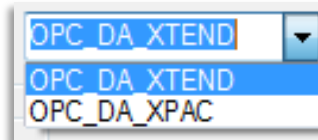
Answer Format Select Box

Server Type

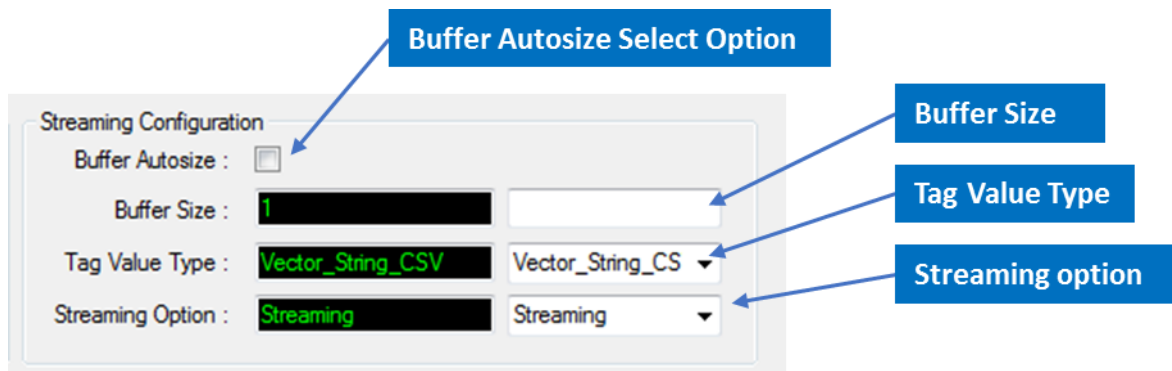
- ✓ **Server ID** : The server ID is a single identifier dedicated to the OPC DA service. For those who are more familiar with COM programming, this identifier is also called the “**CLSID**” . The “**CLSID**” is made up of 36 characters where each character is represented by “X” and is of alphanumeric type. {XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX }
- ✓ **Prog ID** : The prog ID is an alias dedicated to the server ID. The “**Prog ID**” will be visible to OPC DA clients. Its maximum size is limited to 50 characters, and its minimum size to 4 characters. Each character is of alphanumeric type that contains « . » as separator.
- ✓ **Answer Format** : The response format is associated to the response messages sent following the execution of any command from the “Command” tag(cf. sec 3.3.6 and chap. 4). The XML option allows the writing of response messages in XML format and the « **Str** » does the same but in plain text (cf. sec 3.3.6 et chap. 4).



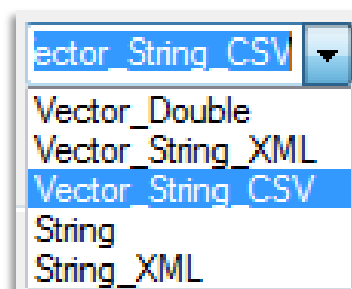
- ✓ **Server Type** : The server type can be chosen between the two following values “OPC_DA_XTEND” and “OPC_DA_XPAC”.



6.2.2 Streaming Configuration



- ✓ **Buffer Size**: The buffer size equals the amount of measurement values that will be set into the tag/item buffer. Each time the amount of transmitted observations exceeds the buffer size, then all data are sent to the item/tag and its value (Double[],String[..]) refreshed. If the Buffer Size AutoSize checkbox is checked then the buffer size will be automatically set to the sampling frequency for each and every sensor.
- ✓ **Tag Value Type**: the tag value type can be set by selecting the desired value from the drop down select box containing the following options “Vector_Double”, “Vector_String_XML”, “Vector_String_CSV”, “String, String_XML”

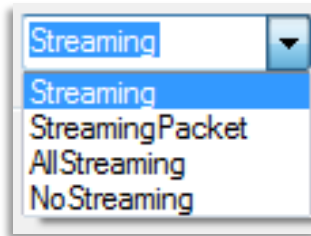


Option Name	Type	Value
Vector_Double	Double[]	For each row: Row Value = Measurement value in Double Ex 1,34
Vector_String_XML	String[]	For each row: Row Value = <Record Value="value"> <Date> date</Date> </Record> Ex: <Record Value="1"> <Date> 2012-07-03T11:52:28.9889096Z</Date> </Record>
Vector_String_CSV,	String[]	For each row: Row Value = Measurement value in String = "Value, Date, Millisecond, Ticks, Index" Ex: 1,23;11:47:47;689;634769128676898202;123232322333
String	String	Measurement value in String = <Value1, Date1, Millisecond1, Ticks1, Index1> <Value2, Date2, Millisecond2, Ticks2, Index2> ...<Value3, Date3, Millisecond3, Ticks3, Index3> Ex: <1;12:14:07;173;634769144471731615;123232322333>... <2,45;12:14:07;174;634769144471731616;123232322334>.... <-21;12:14:07;174;634769144471731616;123232322334>....
String_XML	string	Measurement value in String = <Record Value="value1"> <Date> date1</Date> </Record> <Record Value="value2"> <Date>2 date2</Date> </Record> . . <Record Value="value_n"> <Date>2 date_n</Date> </Record>

```

Ex:
<Record Value="13">
  <Date> 2012-07-03T11:52:28.9889096Z</Date>
</Record>
<Record Value="-10,5">
  <Date> 2012-07-03T11:52:29.9889097Z</Date>
</Record>
.
.<Record Value="-0,23">
  <Date> 2012-07-03T12:00:00.98890909X</Date>
</Record>
    
```

- **Streaming Option:** The user can select the streaming option from the drop down select box containing the following options “StreamingPacket”, “AllStreaming”, “NoStreaming”.



in any case if the SSD mode is selected the device will be transmitting data through the Streaming tag.

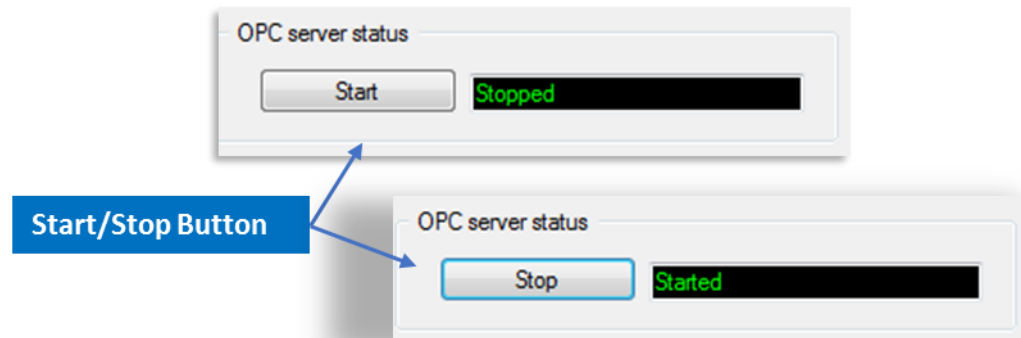
Option	If Selected
StreamingPacket	The Streaming tag will be only dedicated to StreamingPacket mode and SSD transmission
AllStreaming	The Streaming tag will be only dedicated to StreamingPacket mode and SSD transmission
NoStreaming	StreamingPacket modes will be deactivated on all tags (SSD will remain activated)

To validate the previous configuration or to reset all values, you have access Validate/Restore buttons :



- **Validate button:** Validates all input fields « Server ID » and « Prog ID » together with the response format select box through a right click. All input fields left empty will not be impacted.
- **Restore button:** Resets all input fields « Server ID » and « Prog ID » together with the response to their initial factory settings.

6.3 START AND STOP THE OPC SERVER



- **Start the OPC DA server:** Click on the start button, the server estate will then switch on to «Started», the DA OPC server is launched.
- **Stopping the OPC DA server:** Click on the stop button: the server estate will then switch on to «Stopped», the DA OPC server is stopped.

7. DETAILED DATA DESCRIPTION

The OPCA DA splits into two distinct versions, the **OPC DA XPAC** and **OPC DA XTEND**. These two versions differ by the way they display data (items/attributes).

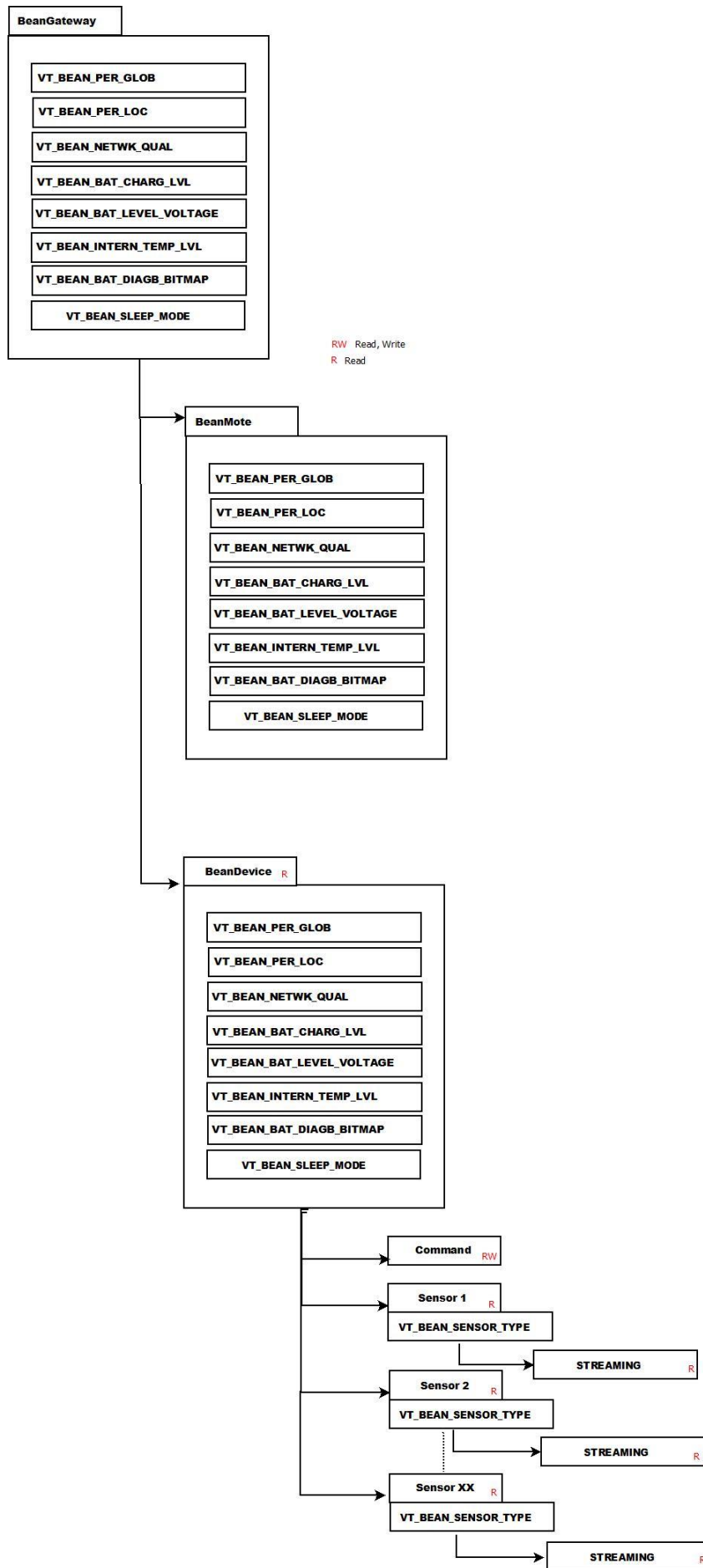
Name	OPC DA XPAC			OPC DA XTEND		
	Attribute ID	Attribute ID	Item/Tag	Attribute ID	Attribute ID	Item/Tag
VT_BEAN_NETWK_QUAL	5001	YES	NO	NC*	NO	YES
VT_BEAN_PER_GLOB	5002	YES	NO	NC*	NO	YES
VT_BEAN_PER_LOC	5003	YES	NO	NC*	NO	YES
VT_BEAN_BAT_LEVEL_VOLTAGE	5004	YES	NO	NC*	NO	YES
VT_BEAN_BAT_CHARG_LVL	5005	YES	NO	NC*	NO	YES
VT_BEAN_BAT_DIAG_BITMAP	5006	YES	NC**	NC*	NO	NC**
VT_BEAN_INTERN_TEMP_LVL	5007	YES	NO	NC*	NO	YES
VT_BEAN_NETWK_DIAG_SEND_FREQ	5008	YES	NC**	NC*	NO	NC**
VT_BEAN_SENSOR_TYPE	5009	YES	NO	NC*	NO	YES
VT_BEAN_SLEEP_MODE	5010	YES	NO	NC*	NO	YES
VT_BEAN_PLATFORM_TYPE	5011	YES	NO	NC*	NO	YES
VT_Acknowledgement	1001	YES	NO	NC*	NO	YES
VT_DownloadStatus	1002	YES	NO	NC*	NO	YES
VT_DataloggerStatus	1003	YES	NO	NC*	NO	YES

NC*=Not concerned

NC**=Not Concerned/ Not implemented

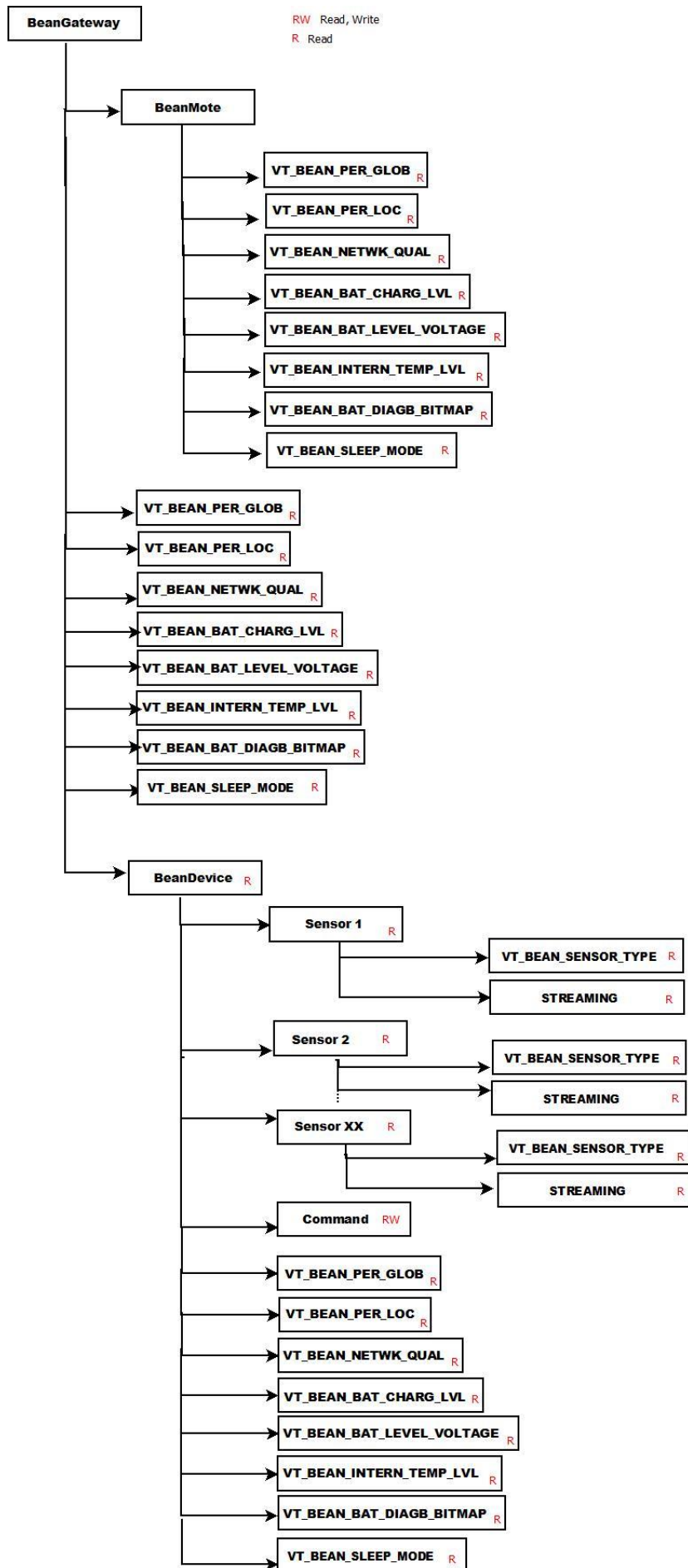
7.1 DATA TREE PRESENTATION: OPC DA XPAC

The **OPC DA XPAC** version represents data under a rather compact data tree which complies to the below drawing. All items bearing the « R » quote are only accessible in read mode. All other items bearing the « RW » quote are accessible in “Read and “Write” mode.



7.2 DATA TREE PRESENTATION: OPC DA XTEND

The DA XTEND version represents data under a rather extended data tree which complies to the below drawing. All items bearing the « **R** » quote are only accessible in read mode. All other items bearing the « **RW** » quote are accessible in “Read” and “Write” mode.



7.3 DATA RELATED TO THE BEANDEVICE

7.3.1 “BeanDevice®” Item/Tag

The following table defines the data acquisition modes applied to all subcomponents which are the sensors.

OPC Name	Value Formatting	Description	Access Right
"DEVICE_" + <PanIdHex> + <MacIdHex>	Byte (unsigned byte) Mode Marker = 0x11 Mode Survey = 0x23	BeanDevice	Read Only (R)

Given the below information:

Gateway_Name="GATEWAY_" + <PanIdHex>

Device_Name=<OPC_Name>

Separator="."

The OPC path determination is as follows:

Address = <Gateway_Name> + <Separator> + <Device_Name>

7.3.2 Sleep Mode/Power Mode attribute OPC DA XPAC

OPC Name	Dynamic/Value	Description	ID Attribut
"VT_BEAN_SLEEP_MODE"	Byte SLEEP_MODE_DISABLED = 0x10 SLEEP_MODE_ENABLED = 0x20 SLEEP_MODE_LISTENING_ENABLED = 0x40 SLEEP_MODE_POWER_OFF = 0x80	Sleep Mode/Power Mode "Sleep Mode"	5010

7.3.3 Sleep Mode/Power Mode Tag/Item OPC DA XTEND

OPC Name	Dynamic/Value	Description	Access Right
"VT_BEAN_SLEEP_MODE"	Byte SLEEP_MODE_DISABLED = 0x10 SLEEP_MODE_ENABLED = 0x20 SLEEP_MODE_LISTENING_ENABLED = 0x40 SLEEP_MODE_POWER_OFF = 0x80	Sleep Mode/Power Mode "Sleep Mode"	Read Only (R)

Given the below information:

Gateway_Name="GATEWAY_"<PanIdHex>

Device_Name="GATEWAY_"<PanIdHex><MacIdHex>

Separator="."

The OPC path determination is as follows:

Address = <Gateway_Name><Separator><Device_Name><Separator><OPC_Name>

7.3.4 Diagnostic Attributes: OPC DA XPAC

OPC Name	Value Formatting	Description	Attribute ID
"VT_BEAN_PER_GLOB"	Float (0 à 10 000 pts)	"Global PER"	5002
"VT_BEAN_PER_LOC"	Float (0 à 10 000 pts)	"Local PER"	5004
"VT_BEAN_NETWK_QUAL"	Uint (0 à 255 pts)	"Network Quality"	5001
"VT_BEAN_BAT_CHARG_LVL"	Float (0 à 10 000 pts)	"Battery Charge Level"	5005
"VT_BEAN_BAT_LEVEL_VOLTAGE"	Float	Tension de la batterie en mV "Battery Voltage Level in mV",	5004
"VT_BEAN_INTERN_TEMP_LVL"	Float	"Battery Internal Temperature"	5007
"VT_BEAN_BAT_DIAGB_BITMAP"	Byte Disable Discharge 0x04 Disable Charge 0x08 Discharge Over Current 0x10 Charge Over Current 0x20 UnderVoltage 0x40 OverVoltage 0x80	"Battery Diagnostic Bitmap"	5006

7.3.5 The Platform Type Attribute OPC D

7.3.6 A XPAC

OPC Name	Value Formatting	Description	Attribute ID
"VT_BEAN_PLATFORM_TYPE"	Byte None = 0, AN_4_20_mA = 0x01; AN_V = 0x02; AN_mV = 0x03; TOR = 0x04; AHD_420 = 0x05; AHD_V = 0x06; AHD_mV = 0x07; AX_3D = 0x10; HPI_2D = 0x11; AX_HD = 0x13; TSI = 0x21; TIR = 0x22; TH = 0x23; SUN_T = 0x24; SUN_IR = 0x25; SUN_TOR = 0x26; SUN_TH = 0x27; SUN = 0x28; SUN_LEVEL = 0x29; TECH_AX_IN = 0x29; TECH_AX_3D_S = 0x2A; TECH_GYR_3D_S = 0x2B;	Sleep Mode/Power Mode "Sleep Mode"	5010

7.3.7 The Platform Type Tag/Item OPC DA XTEND

OPC Name	Value Formatting	Description	Access Right
"VT_BEAN_PLATFORM_TYPE"	Byte None = 0, AN_4_20_mA = 0x01; AN_V = 0x02; AN_mV = 0x03; TOR = 0x04; AHD_420 = 0x05; AHD_V = 0x06; AHD_mV = 0x07; AX_3D = 0x10; HPI_2D = 0x11; AX_HD = 0x13; TSI = 0x21; TIR = 0x22; TH = 0x23; SUN_T = 0x24; SUN_IR = 0x25; SUN_TOR = 0x26; SUN_TH = 0x27; SUN = 0x28; SUN_LEVEL = 0x29; TECH_AX_IN = 0x29;	Sleep Mode/Power Mode "Sleep Mode"	Read Only (R)

	TECH_AX_3D_S = 0x2A; TECH_GYR_3D_S = 0x2B;		
--	---	--	--

Given the below information:

Gateway_Name="GATEWAY_"<PanIdHex>

Device_Name="GATEWAY_"<PanIdHex><MacIdHex>

Separator="."

The OPC path determination is as follows:

Address = <Gateway_Name><Separator><Device_Name><Separator><OPC_Name>

7.3.8 The acknowledgement Tag/Item

This Item represent an information about the last Configuration sent status.

OPC Name	Dynamic/Value	Description	ID Attribut
"VT_Acknowledgement"	Byte CONFIG_MSG_IN_BUFFER = 0x01 CONFIG_MSG_SENT= 0x20 CONFIG_MSG_DELETED = 0x03 CONFIG_MSG_UNDEFINED= 0x04	Last command status	1001

7.3.9 DownloadStatus Item/Tag

OPC Name	Dynamic/Value	Description	ID Attribut
"VT_DownloadStatus"	Byte NONE = 0x00 ANY_DATA_TODOWNLOAD= 0x01 DOWNLOAD_PROCESSING_ = 0x02 DOWNLOAD_COMPLETED= 0x03 DOWNLOAD_CANCELLING= 0x04 DOWNLOAD_CANCELLED= 0x05 DOWNLOAD_TIMEOUT= 0x06 DOWNLOAD_FAILURE= 0x07	Download status	1002

7.3.10 LoggerStatus Item/Tag

OPC Name	Dynamic/Value	Description	ID Attribut
"VT_LoggerStatus"	Byte LOGGER_INITIALIZING = 0x00 LOGGER_READY = 0x01 LOGGER_ACTIVE = 0x02 LOGGER_STOPPED = 0x03 LOGGER_FAILURE = 0x04 LOGGER_NOT_INITIALIZED = 0x05 LOGGER_STOPPEDFULLMEM = 0x06 LOGGER_STAND_ALONE= 0x40	DataLogger Status	1003

7.3.11 Diagnostic Items/Tags: OPC DA XTEND

This table shows all the diagnostics related to the BeanDevice®:

OPC Name	Value Formatting	Description	Access Right
"VT_BEAN_PER_GLOB"	Float (0 à 10 000 pts)	"Global PER"	Read Only (R)
"VT_BEAN_PER_LOC"	Float (0 à 10 000 pts)	"Local PER"	Read Only (R)
"VT_BEAN_NETWK_QUAL"	Uint (0 à 255 pts)	"Network Quality"	Read Only (R)
"VT_BEAN_BAT_CHARG_LVL"	Float (0 à 10 000 pts)	"Battery Charge Level"	Read Only (R)
"VT_BEAN_BAT_LEVEL_VOLTAGE"	Float	"Battery Voltage Level in mV",	Read Only (R)
"VT_BEAN_INTERN_TEMP_LVL"	Float	"Battery Internal Temperature"	Read Only (R)
"VT_BEAN_BAT_DIAGB_BITMAP"	Byte Disable Discharge 0x04 Disable Charge 0x08 Discharge Over Current 0x10 Charge Over Current 0x20 UnderVoltage 0x40 OverVoltage 0x80	Diagnostic Batterie "Battery Diagnostic Bitmap"	Read Only (R)

Given the below information:

Gateway_Name="GATEWAY_" + <PanIdHex>

Device_Name=<OPC_Name>

Separator="."

The OPC path determination is as follows:

Address = <Gateway_Name>+<Separator>+<Device_Name>

7.3.11.1 How to convert and display the diagnostics data

Global Packet Error Rate: Units in % (precision: 3 digits after decimal comma)

$PER_Global_% = PER_Global_Nb_Pts/100$

Local Packet Error Rate: Units in % (precision: 3 digits after decimal comma)

$PER_Local_% = PER_Local_Nb_Pts/100$

Radio signal quality : Units in LQI (Link Quality Indicator)

No conversion

Battery Charge level : Units in % (precision: 3 digits after decimal comma)

$Charge_Lvl\% = Charge_Lvl_Nb_Pts/100$

Battery Voltage : Units in Volts (precision: 3 digits after decimal comma)

$Voltage = 4,88 * voltage_Nb_points/1000$

Internal Temperature : Units in °C (precision: 3 digits after decimal comma)

$Temp_°C = 0,125 * Temp_Nb_points/1000$

7.3.12 The « Command » Item/Tag

OPC Name	Value Formatting	Description	Access Right
"Command"	String	Command chain described below	Read,Write (R,W)

Given the below information:

Gateway_Name="GATEWAY_" + <PanIdHex>

Device_Name="GATEWAY_" + <PanIdHex> + <MacIdHex>

Separator="."

The OPC path determination is as follows:

Address = <Gateway_Name>+<Separator>+<Device_Name>+<Separator>+<OPC_Name>

Generic commands related to measure mode change:

The available commands are the following:

SetMeasureMode *measureMode measureCycle measureDuration measureFrequency*

SetMeasureMode_Survey *measureCycle measureDuration*

Ex: SetMeasureMode_Survey 001,10:10:20 001,00:00:20

SetMeasureMode_LowDutyCycle *measureCycle*

Ex: SetMeasureMode_LowDutyCycle 001,10:10:20

Parameter	Description
measureMode	Value can be as following: "LowDutyCycle" "Alarm" "Paquet_One_Shot" "Paquet" "Math_One_Shot" "Math" "Logger" "Telemetry" "Tracking_Alarm" "Streaming" "Streaming_Alarm" "Survey"
measureCycle	Date value= "DDD, hh:mm:ss" Min Value=1 Day Max Value=365 Days
measureDuration	Value expressed in Seconds
measureFrequency	Valeur in Hz Indicates the sensor sampling rate

7.4 DATA RELATED TO THE SENSOR

7.4.1 "STREAMING" Item/Tag

The "Streaming" TAG provides data related to streaming and SSD data transmission modes. Whenever a device is switched to "Streaming Packet" or "SSD" mode the Streaming tag/item is updated given the chosen data format and size (see section 2.3 OPC configuration).

OPC Name	Value Formatting	Description	Access Right
"Streaming"	Vector_Double Vector_String_XML Vector_String_CSV String String_XML (for more detail refer to section 2.3)	Streaming	Read only (R)

Given the below information:

Gateway_Name="GATEWAY_" + <PanIdHex>

Device_Name="GATEWAY_" + <PanIdHex> + <MacIdHex>

Sensor_Name=<Gateway_Name> + <Separator> + <Device_Name> + <Separator

Separator="."

The OPC path determination is as follows:

Address=

Sensor_Name+"SENSOR_"+<SensIdPrimHex>+"SENSOR_"+<SensIdPrimHex>+<OPC_Name>

7.4.2 “BEANSENSOR” Item/Tag

The BeanSensor® technically provides data measurement as well as other information.

OPC Name	Value Formatting	Description	Access Right
"SENSOR_"+<SensIdPrimHex>	Double In case of the padlock proceeded to the below rule. Rule: In order to obtain a bitmap, procede to unsigned Byte Conversion	BeanSensor	Read only (R)

Given the below information:

Gateway_Name="GATEWAY_"+<PanIdHex>

Device_Name="GATEWAY_"+<PanIdHex>+<MacIdHex>

Separator="."

The OPC path determination is as follows:

Address = <Gateway_Name>+<Separator>+<Device_Name>+<Separator>+<OPC_Name>

7.4.3 « Type » attribute: OPC DA XPAC

The Beansensor type is communicated through the «VT_BEAN_SENSOR_TYPE» attribute associated to each «Beansnsor».

OPC Name	Value Formatting	Description	Attribute ID
"VT_BEAN_SENSOR_TYPE"	byte AN_4_20_mA = 0x01; AN_V = 0x02; AN_mV = 0x03; INT_TEMP = 0x04; AX_3D = 0x05; HPI_2D = 0x06; AX_HD = 0x07; AHD_420 = 0x08; AHD_V = 0x09; AHD_mV = 0x0A; TSI = 0x0F; TOR = 0x15; CADENAS = 0x16;	Sensor technology	5009

7.4.4 « Type » Item/Tag: OPC DA XTEND

The Beansensor type is communicated through the «VT_BEAN_SENSOR_TYPE» Item/Tag associated to each « Beansnsor ».

OPC Name	Value Formatting	Description	Access Right
“VT_BEAN_SENSOR_TYPE”	Byte AN_4_20_mA = 0x01; AN_V =0x02; AN_mV = 0x03; INT_TEMP = 0x04; AX_3D =0x05; HPI_2D = 0x06; AX_HD = 0x07; AHD_420 = 0x08; AHD_V = 0x09; AHD_mV = 0x0A; TSI = 0x0F; TOR = 0x15; CADENAS = 0x16;	Type technologie du capteur	Lecture Seule (R)

Given the below information:

Gateway_Name="GATEWAY_"<PanIdHex>

Device_Name="GATEWAY_"<PanIdHex><MacIdHex>

Sensor_Name="SENSOR_"<SensIdPrimHex>

Separator="."

The OPC path determination is as follows:

Address = <Gateway_Name><Separator><Device_Name><Separator>< Sensor_Name >< Separator ><OPC_Name>

8. DETAILED DESCRIPTION OF BENGATEWAY COMMANDS

The execution of each command from the “Command” item attached to the “BeanGateway” Item leads both to a system response « MasterError » and the writing of a response message on the “Command” Item/Tag under “XML” or “Str” format (Configurable from the GUI).

8.1 SYSTEM ACKNOWLEDGMENT

The « MasterError » is systematically updated after each command execution

Acknowledgment Code	Description
<i>Fail</i>	Command not executed
<i>Ok</i>	Command executed

8.2 COMMANDS FOR CONFIGURING THE GATEWAY

8.2.1 The “SetClockBoradcastCycle” Command

This command sets the broadcasting cycle to the desired value. (refer to the Beanscape® user manual).

8.2.1.1 Command Syntax

```
SetClockBoradcastCycle clockBroadcastingCycle
Ex: SetClockBoradcastCycle 20
```

Parameters	Description
clockBroadcastingCycle	Value type (UInt32) Min Value=0 Max Value= (refer to the Beanscape® user manual)

8.2.1.2 *Response Syntax*

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> • « OK » if command has been executed • « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

8.2.2 The “SetNetwkDiagCycle” Command

This command sets the network diagnostics cycle to the desired value. (refer to the Beanscape® user manual).

8.2.2.1 *Command Syntax*

```
SetNetwkDiagCycle networkDiagCycleConf
Ex: SetNetwkDiagCycle 120
```

Parameters	Description
networkDiagCycleConf	Value type (UInt32) Min Value=0 Max Value= (refer to the Beanscape® user manual)

8.2.2.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

8.2.3 The “SetTxPowerConfig” Command

This command sets the power configuration to the desired value. (refer to the Beanscape® user manual).

8.2.3.1 Command Syntax

```
SetTxPowerConfig networkDiagCycleConf
Ex: SetNetwkDiagCycle 0
```

Parameters	Description
networkDiagCycleConf	Value type (UInt32) Min Value = 0 Max Value= 5 Mapping: 0 = -7 dBm 1 = -1 dBm 2 = +5 dBm 3 = +11 dBm 4 = +15 dBm 5 = +18 dBm

Response Syntax

<i>Response Parameter</i>	<i>Parameter</i>	<i>Description</i>
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

8.2.4 The “EraseGatewayNetworkContext” Command

This command erases the BeanGateway®’s network context. (refer to the Beanscape® user manual).

8.2.4.1 *Command Syntax*

```
EraseGatewayNetworkContext networkDiagCycleConf
Ex: EraseGatewayNetworkContext 0
```

Parameters	Description
clockBroadcastingCycle	Value type (UInt16) Min Value = 0 Max Value= 1 Mapping: 0 = Configuration Reset (“Carto”) 1 = Profile Reset (“Full”)

Response Syntax

<i>Response Parameter</i>	<i>Parameter</i>	<i>Description</i>
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> • « OK » if command has been executed • « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```

<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>

```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9. DETAILED DESCRIPTION OF BEANDEVICE COMMANDS

The execution of each command from the “Command” item attached to the “BeanDevice” Item leads both to a system response « MasterError » and the writing of a response message on the “Command” Item/Tag under “XML” or “Str” format (Configurable from the GUI).

9.1 SYSTEM ACKNOWLEDGMENT

The « MasterError » is systematically updated after each command execution

Acknowledgment Code	Description
<i>Fail</i>	Command not executed
<i>Ok</i>	Command executed

9.2 COMMANDS FOR SETTING THE MEASURE MODE

9.2.1 The “SetMeasureMode_Survey” Command

This command sets the sensor measure mode to “Survey” mode. The logger mode will be set to “TX Only” which means that the data will be only transmitted to the gateway and not recorded within the device flash memory (refer the Beanscape user manual).

9.2.1.1 Command Syntax

```
SetMeasureMode_Survey measureCycle measureCycleDuration
Ex: SetMeasureMode_Survey 000,00:00:10 001,00:00:20
```

Parameters	Description
measureCycle	Date value= “DDD, hh:mm:ss” Min Value=1 Day Max Value=365 Days
measureCycleDuration	Date value= “DDD, hh:mm:ss” Min Value=1 Day Max Value=365 Days

```
Rule : measureCycleDuration <= measureCycle
      AND
      measureCycleDuration % measureCycle != 0
```

The measurement cycle duration is less or equal to the measure cycle and the measure cycle is a duration acquisition cycle multiple

9.2.1.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.2.2 The “SetMeasureMode_Survey_WLogger” Command

This command sets the sensor measure mode to “Survey” mode. The logger mode will be set to the desired option.

9.2.2.1 Command Syntax

```
SetMeasureMode_Survey_WLogger measureCycle measureCycleDuration logger_option
Ex: SetMeasureMode_Survey_WLogger 000,00:00:10 001,00:00:20 2
```

Parameters	Description
measureCycle	Date value= “DDD, hh:mm:ss” Min Value=1 Day Max Value=365 Days
measureCycleDuration	Date value= “DDD, hh:mm:ss” Min Value=1 Day Max Value=365 Days
Logger_option	If value is set to “0” then the transmission option will be set to “TX” If value is set to “1” then the transmission option will be set to “Log” If value is set to “2” then the transmission option will be set to “Log&TX” If Else then the transmission option will be set to “TX”

```
Rule : measureCycleDuration <= measureCycle
      AND
      measureCycleDuration % measureCycle != 0
```

The measurement cycle duration is lower or equal to the measure cycle and the measure cycle is a duration acquisition cycle multiple

9.2.2.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.2.3 The “SetMeasureMode_LowDutyCycle” Command

This command sets the sensor measure mode to “LowDutyCycle” mode. The logger mode will be set to “TX Only” which means that the data will be only transmitted to the gateway and not recorded within the device flash memory (refer the Beanscape user manual).

9.2.3.1 Command Syntax

```
SetMeasureMode_LowDutyCycle measureCycle
Ex: SetMeasureMode_LowDutyCycle 001,10:10:20
```

Parameter	Description
measureCycle	Date value= “DDD, hh:mm:ss” Min Value=1 Day Max Value=365 Days

9.2.3.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.2.4 The “SetMeasureMode_LowDutyCycle_WLogger” Command

This command sets the sensor measure mode to “LowDutyCycle” mode. The logger mode will be set to the desired option.

9.2.4.1 Command Syntax

```
SetMeasureMode_LowDutyCycle_Wlogger measureCycle logger_option
Ex: SetMeasureMode_LowDutyCycle_WLogger 001,10:10:20 3
```

Parameter	Description
measureCycle	Date value= “DDD,hh:mm:ss” Min Value=1 Day Max Value=365 Days
Logger_option	If value is set to “0” then the transmission option will be set to “TX” If value is set to “1” then the transmission option will be set to “Log” If value is set to “2” then the transmission option will be set to “Log&TX” If Else then the transmission option will be set to “TX”

9.2.4.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.2.5 The “SetMeasureMode_StreamingPacket” Command

This command sets the sensor measure mode to “StreamingPacket” mode. The logger mode will be set to the desired option.

9.2.5.1 Command Syntax

```
SetMeasureMode_StreamingPacket measureFrequency measureCycle
measureCycleDuration logger_option
```

Ex: *SetMeasureMode_StreamingPacket 1000 000,00:00:10 001,00:00:20 2*

Parameters	Description
measureFrequency	Frequency in Hz Min Value=1 Hz
measureCycle	Date value= “DDD, hh:mm:ss” Min Value=1 Day Max Value= see sensor specification
measureCycleDuration	Date value= “DDD, hh:mm:ss” Min Value=1 Day Max Value=365 Days
Logger_option	If value is set to “0” then the transmission option will be set to “TX” If value is set to “1” then the transmission option will be set to “Log” If value is set to “2” then the transmission option will be set to “Log&TX” If Else then the transmission option will be set to “TX”

```
Rule : measureCycleDuration <= measureCycle
      AND
      measureCycleDuration % measureCycle != 0
```

The measurement cycle duration is lower or equal to the measure cycle and the measure cycle is a duration acquisition cycle multiple

9.2.5.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.2.6 The “SetMeasureMode_StreamingPacket_CntMon” Command

This command sets the sensor measure mode to “StreamingPacket” mode with the continuous streaming mode turned set to “ON”. The logger mode will be set to the desired option.

9.2.6.1 Command Syntax

```
SetMeasureMode_StreamingPacket_CntMon measureFrequency logger_option
Ex: SetMeasureMode_StreamingPacket_CntMon 600 2
```

Parameters	Description
measureFrequency	Frequency in Hz Min Value=1 Hz
Logger_option	If value is set to “0” then the transmission option will be set to “TX” If value is set to “1” then the transmission option will be set to “Log” If value is set to “2” then the transmission option will be set to “Log&TX” If Else then the transmission option will be set to “TX”

9.2.6.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.2.7 The “SetMeasureMode_Streaming_CntMon” Command

This command sets the sensor measure mode to “Streaming” mode with the continuous streaming mode set to “ON” . The logger mode will be set to the desired option.

9.2.7.1 Command Syntax

```
SetMeasureMode_Streaming_CntMon measureFrequency logger_option
Ex: SetMeasureMode_StreamingPacket_CntMon 100 0
```

Parameters	Description
measureFrequency	Frequency in Hz Min Value=1 Hz
Logger_option	If value is set to “0” then the transmission option will be set to “TX” If value is set to “1” then the transmission option will be set to “Log” If value is set to “2” then the transmission option will be set to “Log&TX” If Else then the transmission option will be set to “TX”

9.2.7.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.2.8 The “SetMeasureMode_SSD” Command

This command sets the sensor measure mode to “SSD” mode. The logger mode will be set to the desired option.

9.2.8.1 Command Syntax

```
SetMeasureMode_SSD measureFrequency measureCycle measureCycleDuration
logger_option
Ex: SetMeasureMode_SSD 100 000,00:00:10 001,00:00:20 0
```

Parameters	Description
measureFrequency	Frequency in Hz Min Value=1 Hz
measureCycle	Date value= “DDD, hh:mm:ss” Min Value=1 Day Max Value= see sensor specification
measureCycleDuration	Date value= “DDD, hh:mm:ss” Min Value=1 Day Max Value=365 Days
Logger_option	If value is set to “0” then the transmission option will be set to “TX” If value is set to “1” then the transmission option will be set to “Log” If value is set to “2” then the transmission option will be set to “Log&TX” If Else then the transmission option will be set to “TX”

```
Rule : measureCycleDuration <= measureCycle
      AND
      measureCycleDuration % measureCycle != 0
```

The measurement cycle duration is lower or equal to the measure cycle and the measure cycle is a duration acquisition cycle multiple

9.2.8.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.2.9 The “SetMeasureMode_SSD_CntMon” Command

This command sets the sensor measure mode to “SSD” mode with the continuous streaming mode set to “ON” . The logger mode will be set to the desired option.

9.2.9.1 Command Syntax

```
SetMeasureMode_SSD measureFrequency logger_option
Ex: SetMeasureMode_SSD 100 0
```

Parameters	Description
measureFrequency	Frequency in Hz Min Value=1 Hz
logger_option	If value is set to “0” then the transmission option will be set to “TX” If value is set to “1” then the transmission option will be set to “Log” If value is set to “2” then the transmission option will be set to “Log&TX” If Else then the transmission option will be set to “TX”

9.2.9.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.2.10 The “SetMeasureMode_StreamingPacket_Burst” Command

This Command set the sensor measure mode to “StreamingPacket” mode and configure the measurement cycle and the measurement duration.

9.2.10.1 Command Syntax

```
SetMeasureMode_StreamingPacket_Burst measureFrequency measureCycle
measureCycleDuration logger_option
Ex: SetMeasureMode_StreamingPacket_Burst 1000 000,00:00:10 001,00:00:20 2
```

Parameters	Description
measureFrequency	Frequency in Hz Min Value=1 Hz
measureCycle	Date value= “DDD, hh:mm:ss” Min Value=1 Day Max Value= see sensor specification
measureCycleDuration	Date value= “DDD, hh:mm:ss” Min Value=1 Day Max Value=365 Days
logger_option	If value is set to “0” then the transmission option will be set to “TX” If value is set to “1” then the transmission option will be set to “Log” If value is set to “2” then the transmission option will be set to “Log&TX” If Else then the transmission option will be set to “TX”

```
Rule : measureCycleDuration <= measureCycle
      AND
      measureCycleDuration % measureCycle != 0
```

The measurement cycle duration is lower or equal to the measurement cycle and the measurement cycle is a duration acquisition cycle multiple

9.2.10.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

```
Str_ReturnCode : Str_Response: Dt_Date
```

9.2.11 The “SetMeasureMode_StreamingPacket_Burst” Command

This Command set the sensor measure mode to “Streaming” mode and configure the measurement cycle, the measurement duration and the datalogger option.

9.2.11.1 Command Syntax

```
SetMeasureMode_Streaming_Burst measureFrequency measureCycle
measureCycleDuration logger_option
Ex: SetMeasureMode_StreamingPacket_Burst 1000 000,00:00:10 001,00:00:20 2
```


Parameters	Description
measureFrequency	Frequency in Hz Min Value=1 Hz Max Value=100 Hz
measureCycle	Date value= "DDD, hh:mm:ss" Min Value=1 Day Max Value= see sensor specification
measureCycleDuration	Date value= "DDD, hh:mm:ss" Min Value=1 Day Max Value=365 Days
logger_option	If value is set to "0" then the transmission option will be set to "TX" If value is set to "1" then the transmission option will be set to "Log" If value is set to "2" then the transmission option will be set to "Log&TX" If Else then the transmission option will be set to "TX"

```
Rule : measureCycleDuration <= measureCycle
      AND
      measureCycleDuration % measureCycle != 0
```

The measurement cycle duration is lower or equal to the measure cycle and the measure cycle is a duration acquisition cycle multiple

9.2.11.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

```
Str_ReturnCode : Str_Response: Dt_Date
```

9.2.12 The “SetMeasureMode_StreamingPacket_OneShot” Command

This Command set the sensor measure mode to “StreamingPacket” mode and configure the measurement duration and the datalogger option.

9.2.12.1 Command Syntax

```
SetMeasureMode_StreamingPacket_OneShot measureFrequency MaeasureDuration
logger_option
Ex: SetMeasureMode_StreamingPacket_OneShot 1000 000,00:00:10 2
```

Parameters	Description
measureFrequency	Frequency in Hz Min Value=1 Hz See sensor specifaicon
measureDuration	Date value= “DDD, hh:mm:ss” Min Value=1 Day Max Value=365 Days
Logger_option	If value is set to “0” then the transmission option will be set to “TX” If value is set to “1” then the transmission option will be set to “Log” If value is set to “2” then the transmission option will be set to “Log&TX” If Else then the transmission option will be set to “TX”

9.2.12.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

```
Str_ReturnCode : Str_Response: Dt_Date
```

9.2.13 The “SetMeasureMode_StreamingPacket_OneShot” Command

This Command set the sensor measure mode to “Streaming” mode and configuring the measurement duration and the datalogger option.

9.2.13.1 Command Syntax

```
SetMeasureMode_Streaming_OneShot measureFrequency MaeasureDuration
logger_option
Ex: SetMeasureMode_StreamingPacket_OneShot 100 000,00:00:10 2
```

Parameters	Description
measureFrequency	Frequency in Hz Min Value=1 Hz Max Value=100 HZ
measureDuration	Date value= “DDD, hh:mm:ss” Min Value=1 Day Max Value=365 Days
Logger_option	If value is set to “0” then the transmission option will be set to “TX” If value is set to “1” then the transmission option will be set to “Log” If value is set to “2” then the transmission option will be set to “Log&TX” If Else then the transmission option will be set to “TX”

9.2.13.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

```
Str_ReturnCode : Str_Response: Dt_Date
```

9.2.14 The “SetMeasureMode_Commissioning” Command

This command set the sensor Measure mode to “Commissioning” mode

9.2.14.1 Command Syntax

SetMeasureMode_Commissioning *Timeout*
Ex: SetMeasureMode_Commissioning 600

Parameters	Description
Timeout	Min Value=60 s Max Value=3600 s

9.2.14.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> • « OK » if command has been executed • « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : **Str_Response**: **Dt_Date**

9.3 COMMANDS FOR CONFIGURING THE DIGNOSTICS AND TX POWER MODE

9.3.1 The “SetNetwkDiagCycle” command

This command sets the device diagnostics cycles to the desired frequency

9.3.1.1 Command Syntax

```
SetNetwkDiagCycle diagnosticCycle
```

Ex: *SetNetwkDiagCycle 100*

Parameters	Description
diagnosticCycle	Frequency (UInt32) Min Value= see device specifications Max Value= see device specifications

9.3.1.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.3.2 The “SetTxPowerConfig” command

This command sets the device power configuration desired option

9.3.2.1 Command Syntax

SetTxPowerConfig *powerOption*

Ex: SetNetwkDiagCycle 1

Parameters	Description
powerOption	Option (Byte) Min Value= 0 Max Value= 5

9.3.2.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> • « OK » if command has been executed • « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : **Str_Response**: **Dt_Date**

9.4 LEAVE NETWORK COMMAND

This command issues a "LeaveNetwork" command.

9.4.1.1 Command Syntax

LeaveNetwork

Ex: LeaveNetwork

9.4.1.2 Response Syntax

<i>Response Parameter</i>	<i>Parameter</i>	<i>Description</i>
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> • « OK » if command has been executed • « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.5 COMMANDS FOR CONFIGURING THE SLEEP MODE

9.5.1 The “SetWaitingFrameDeletion” command

This command issues a “waitFrameDeletion” command.

9.5.1.1 Command Syntax

SetWaitingFrameDeletion

Ex: SetWaitingFrameDeletion

9.5.1.2 Response Syntax

<i>Response Parameter</i>	<i>Parameter</i>	<i>Description</i>
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> • « OK » if command has been executed • « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.5.2 The “EnableSleepMode” command

This command sets the “Sleep Mode” to “ON”.

9.5.2.1 Command Syntax

EnableSleepMode

Ex: *EnableSleepMode*

9.5.2.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> • « OK » if command has been executed • « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.5.3 The “DisableSleepMode” command

This command sets the “sleep mode” to “OFF”.

9.5.3.1 Command Syntax

DisableSleepMode

Ex: *DisableSleepMode*

9.5.3.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> • « OK » if command has been executed • « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.5.4 The “EnableSleepWithListening” command

This command enables the “Sleep Mode” with the listening option turned “ON”.

9.5.4.1 Command Syntax

EnableSleepWithListening *ratio*

Ex: *EnableSleepWithListening 50*

Parameters	Description
powerOption	Option (uint16) Min Value= 1 Max Value=100

9.5.4.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> • « OK » if command has been executed • « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.6 COMMANDS FOR CONFIGURING THE DATA LOGGER

9.6.1 “Logger_Download_Data” command

This command issues a download request to the device.

9.6.1.1 Command Syntax

```
Logger_Download_Data
```

Ex: *Logger_Download_Data*

9.6.1.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> • « OK » if command has been executed • « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.6.2 “Logger_Stop_Logging_Data” command

This command issues a stop logging command order to the device.

9.6.2.1 Command Syntax

```
Logger_Stop_Logging_Data
Ex: Logger Stop Logging Data
```

9.6.2.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.6.3 “Logger_Erase_Stored_Log” command

This command issues an erase stored log command order to the device.

9.6.3.1 Command Syntax

Logger_Erase_Stored_Log
 Ex: *Logger_Erase_Stored_Log*

9.6.3.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> • « OK » if command has been executed • « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.6.4 “Logger_Cancel_Upload” command

This command issues a cancel upload stored log request to the device.

9.6.4.1 Command Syntax

Logger_Cancel_Upload
 Ex: *Logger_Cancel_Upload*

9.6.4.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> • « OK » if command has been executed • « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.6.5 DownloadThenErase Command

This command is used to download then erase the logged data.

9.6.5.1 Command Syntax

DownloadThenErase
Ex: DownloadThenErase

9.6.5.2 Response Syntax

<i>Response Parameter</i>	<i>Parameter</i>	<i>Description</i>
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.6.6 SCDE Command

This command is used to set the DeanDevice in commissioning mode the download logger data then erase logged data .

9.6.6.1 Command Syntax

SCDE
Ex: SCDE

9.6.6.2 Response Syntax

Response Parameter	Parameter	Description
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.7 DATA LOGGER FULL MEMORY STRATEGY COMMANDS:

9.7.1 "SetFullMemory_SC" command :

This command sets the DdataLogger full memory strategy to "Switch to commissioning".

9.7.1.1 Command Syntax

```
SetFullMemory_SC
```

Ex: *SetFullMemory_SC*

9.7.1.2 Response Syntax

<i>Response Parameter</i>	<i>Parameter</i>	<i>Description</i>
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.7.2 “SetFullMemory_SAE” command :

This command sets the DdataLogger full memory strategy to “Stop at the end”.

9.7.2.1 Command Syntax

SetFullMemory_SAE

Ex: *SetFullMemory SAE*

9.7.2.2 Response Syntax

<i>Response Parameter</i>	<i>Parameter</i>	<i>Description</i>
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> « OK » if command has been executed « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

9.7.3 “SetFullMemory_SCDE” command :

This command sets the DdataLogger full memory strategy to “Switch to commissioning, download then erase”.

9.7.3.1 Command Syntax

```
SetFullMemory_SCDE
```

Ex: *SetFullMemory_SCDE*

9.7.3.2 Response Syntax

<i>Response Parameter</i>	<i>Parameter</i>	<i>Description</i>
Return Code	Str_ReturnCode	Return Code <ul style="list-style-type: none"> • « OK » if command has been executed • « NOK » if command has not been executed
Request	Str_Request	The executed command chain with all its parameters
Response	Str_Response	Response message following the command execution
Date	Dt_Date	Command execution date

Syntax XML

```
<Message Date="Dt_Date">
  <Request>Str_Request</Request>
  <Response ReturnCode="Str_ReturnCode">Str_Response</Response>
</Message>
```

Syntax Str

Str_ReturnCode : Str_Response: Dt_Date

10. DETAILED DESCRIPTION OF THE TAG VALUE

The tag value type is a variant as shown on the below table.

.NET Types and Their Marshaling Behavior Inside VARIANTS

Type of Instance	Variant Type
null (Nothing in VB .NET)	VT_EMPTY
System.DBNull	VT_NULL
System.Runtime.InteropServices.CurrencyWrapper	VT_CY
System.Runtime.InteropServices.UnknownWrapper	VT_UNKNOWN
System.Runtime.InteropServices.DispatchWrapper	VT_DISPATCH
System.Runtime.InteropServices.ErrorWrapper	VT_ERROR
System.Reflection.Missing	-VT_ERROR with value DISP_E_PARAMNOTFOUND
System.String	VT_BSTR
System.Decimal	VT_DECIMAL
System.Boolean	VT_BOOL
System.Char	VT_U2
System.Byte	VT_U1
System.SByte	VT_I1
System.Int16	VT_I2
System.Int32	VT_I4
System.Int64	VT_I8
System.IntPtr	VT_INT
System.UInt16	VT_U2
System.UInt32	VT_U4
System.UInt64	VT_U8
System.UIntPtr	VT_UINT
System.Single	VT_R4
System.Double	VT_R8
System.DateTime	VT_DATE
Any Array	VT_... VT_ARRAY
System.Object or other .NET classes	VT_DISPATCH

The tag Value is to and initialized but empty object whenever the value type is unknown “**System.Object**”. Because null (Nothing) is mapped to an "empty object" (a VARIANT with type VT_EMPTY) when passed to COM via a **System.Object** parameter, the new **DispatchWrapper(null)** or new **UnknownWrapper(null)** is passed on to represent a null object. This maps to a VARIANT with type VT_DISPATCH or VT_UNKNOWN, respectively, whose pointer value is null. Every tag will be cast to its definite value before being passed on to the OPC server.

11. DETAILED DESCRIPTION OF STANDARD TAG ATTRIBUTES “DATE” AND “QUALITY”

11.1 DATE ATTRIBUTE

The date attribute on each tag refers to the exact time when the observed value was recorded on a remote device (Beandevic[®]) and not to the date/time when the data was recorded on the OPC server.

11.2 QUALITY ATTRIBUTE

The “quality” attribute is used to give an indication regarding the quality/reliability of the value. If one device is turned off or stops emitting, or if the connection is interrupted, then the item/ attribute “quality” and all its sub items will be set to “BAD” (except the “type” attribute/item). Indeed once the “type” attribute is set, its quality turns to good and will not change throughout time. The reason being that a sensor type will never change and the type value will not be submitted to any doubt whether the network is working or not.

Quality enum table:

<i>Quality</i>	<i>Quality Code</i>	<i>Description</i>
Bad	0	The value is bad but no specific reason is known.
badConfigurationError	4	There is some server specific problem with the configuration. For example the item in question has been deleted from the configuration.
badNotConnected	8	The input is required to be logically connected to something but is not. This quality may reflect that no value is available at this time, for reasons like the value may have not been provided by the data source.
badDeviceFailure	12	A device failure has been detected.
badSensorFailure	16	A sensor failure had been detected (the 'Limits' field can provide additional diagnostic information in some situations).
badLastKnownValue	20	Communications have failed. However, the last known value is available.
badCommFailure	24	Communications have failed. There is no last known value is available.
badOutOfService	28	The block is off scan or otherwise locked. This quality is also used when the active state of the item or the group containing the item is InActive.
badWaitingForInitialData	32	After Items are added to a group, it may take some time for the server to actually obtain values for these items. In such

		cases the client might perform a read (from cache), or establish a ConnectionPoint based subscription and/or execute a Refresh on such a subscription before the values are available.OPC DA 3.0 or newer servers.
uncertain	64	The quality of the value is uncertain. There is no specific reason why the value is uncertain.
uncertainLastUsableValue	68	Whatever was writing this value has stopped doing so. The returned value should be regarded as 'stale'.
uncertainSensorNotAccurate	80	Either the value has 'pegged' at one of the sensor limits (in which case the limit field should be set to 1 or 2) or the sensor is otherwise known to be out of calibration via some form of internal diagnostics (in which case the limit field should be 0).
uncertainEUExceeded	84	The returned value is outside the limits defined for this parameter. Note that in this case (per the Fieldbus Specification) the 'Limits' field indicates which limit has been exceeded but does NOT necessarily imply that the value cannot move farther out of range.
uncertainSubNormal	88	The value is derived from multiple sources and has less than the required number of Good sources.
Good	192	The value is good. There are no special conditions.
goodLocalOverride	216	The value has been Overridden. Typically this is means the input has been disconnected and a manually entered value has been 'forced'.

Algorithm variable table:

Variable	Variable Description
no_transmit_alert	Boolean: True if the no transmit alert has been issued, else value set to False: The alert is issued whenever a no transmission cycle superior to 2 times the measurement cycle has been observed.
gateway_stopped	Boolean: True is gateway stopped, else value set to False
sensor_quality	Enum: (see Quality enum table: section 5.2)
device_quality	Enum: (see Quality enum table: section 5.2)
gateway_quality	Enum: (see Quality enum table: section 5.2)
beanscape_quality	Enum: (see Quality enum table: section 5.2)
device_sleepmode	SLEEP_MODE_DISABLED SLEEP_MODE_ENABLED SLEEP_MODE_LISTENING_ENABLED SLEEP_MODE_POWER_OFF SLEEP_MODE_STANDBY
power_Off	Boolean: True if device turned off, False if else

sensor_switch_On	Boolean: True if switched on, False if else
beanscape_server_stopped	Boolean: True if the server is switched Off, False if else

11.2.1 Sensor Quality Determination

The sensor quality algorithm:

```

IF ( no_transmit_alert == true OR gateway_stopped==true)
  THEN sensor_quality= badCommFailure
  EXIT
ENDIF

```

```

IF (device_sleepmode == power_Off)
  THEN sensor_quality= badNotConnected
ELSE
  THEN
    IF sensor_switch_On=true
      then then sensor_quality=good
    ELSE
      then sensor_quality= badNotConnected
    ENDIF
  ENDIF
ENDIF

```

The same rule applies to all sub attribute TAG quality. The below tag quality attributes quality automatically inherit from the sensor signal quality attributes (in case of the OPC_XTEND version)

Sub tag list (OPC_XTEND case):

VT_BEAN_SENSOR_TYPE

11.2.2 BeanDevice® Quality Determination

The device quality algorithm:

```
IF ( no_transmit_alert == true OR gateway_stopped==true)
  THEN device_quality= badCommFailure
  EXIT
ENDIF

IF (device_sleepmode == power_Off)
  THEN device_quality= badNotConnected
ELSE
  THEN device_quality= good
ENDIF
```

The same rule applies to all sub attribute TAG quality. The below tag quality attributes quality automatically inherit from the sensor signal quality attributes (in case of the OPC_XTEND version).

Sub tag list (OPC_XTEND case):

```
VT_BEAN_BAT_CHARG_LVL
VT_BEAN_BAT_DIAG_BITMAP
VT_BEAN_BAT_LEVEL_VOLTAGE
VT_BEAN_INTERN_TEMP_LVL
T_BEAN_NETWK_QUAL
VT_BEAN_PER_GLOB
VT_BEAN_PER_LOC
VT_BEAN_SLEEP_MODE
```

11.2.3 BeanGateway® Quality Determination

The device quality algorithm:

```
IF ( gateway_stopped==true)
  THEN gateway_quality=bad_comfailure
ELSE
  THEN gateway_quality=good
ENDIF
```

The same rule applies to all sub attribute TAG quality. The below tag quality attributes quality automatically inherit from the sensor signal quality attributes (in case of the OPC_XTEND version)

Sub tag list (OPC_XTEND case):

```
VT_BEAN_BAT_CHARG_LVL
VT_BEAN_BAT_DIAG_BITMAP
VT_BEAN_BAT_LEVEL_VOLTAGE
VT_BEAN_INTERN_TEMP_LVL
VT_BEAN_NETWK_QUAL
VT_BEAN_PER_GLOB
VT_BEAN_PER_LOC
VT_BEAN_SLEEP_MODE
```

11.2.4 BeanScope® Quality Determination

The Beandevic[®] quality algorithm:

```
IF ( beanscape_Server_stopped==true)
  THEN beanscape_quality=bad_comfailure
ELSE
  THEN gateway_quality=good
ENDIF
```

11.2.5 Command tag quality determination

The command tag quality attribute will be set to “good” as long as the OPC server’s switched “ON”

12. APPENDICES

12.1 APPENDICE1: FIRST STEPS TO DEVELOP YOUR OPC CLIENT WITH C#



Download the whole code source of OPC DA project optimized by BeanAir® from <http://www.beanair.com/opc-toolkit-wireless-iot-sensors.html>



The source code was deployed using “OPCdotNETLib” library



OPC Core Components Redistributable is mandatory to work with OPC.

OPC servers listing

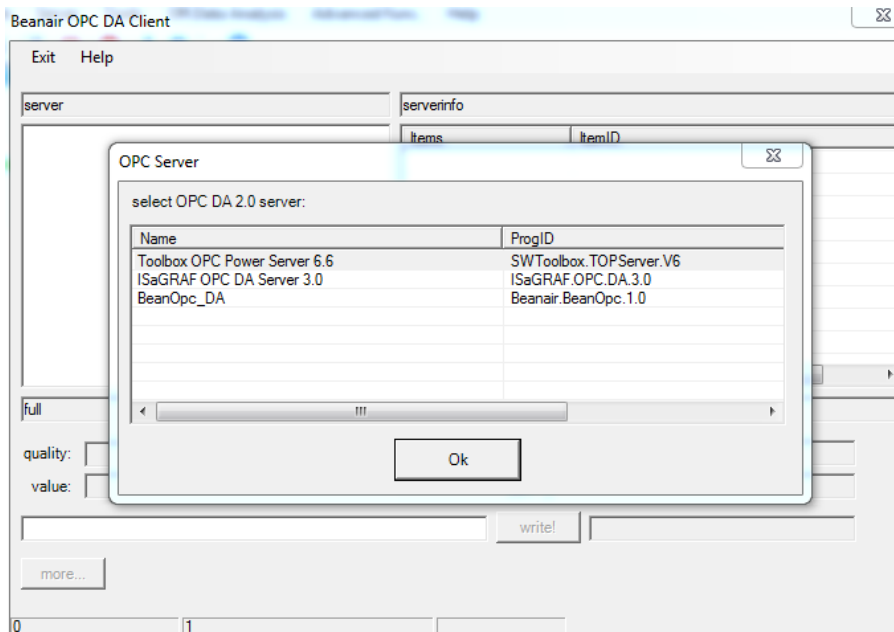
```
public void ListAllServers()
{
    OpcDaSrvList.Items.Clear();

    OpcServerList lst = new OpcServerList();
    OpcServers[] svcs = null;
    try
    {
        lst.ListAllData20( out svcs );
    }
    catch( COMException )
    {
        MessageBox.Show( this, "Enum OPC servers failed!", "Select Server",
        MessageBoxButtons.OK, MessageBoxIcon.Warning );
        return;
    }

    if( svcs == null )
        return;

    string[] itemstrings = new string[ 3 ];
    foreach( OpcServers l in svcs )
    {
        itemstrings[0] = l.ServerName;
        itemstrings[1] = l.ProgID;
        itemstrings[2] = l.ClsID.ToString();
        OpcDaSrvList.Items.Add( new ListViewItem( itemstrings ) );
    }

    // preselect top item in ListView
    OpcDaSrvList.Items[0].Selected = true;
}
```



Connecting to a selected OPC server

```
// connect to OPC server via ProgID
public bool DoConnect( string progid )
{
    try
    {
        theSrv.Connect( progid );
        Thread.Sleep( 100 );
        theSrv.SetClientName( "DirectOPC " + thisprocess.Id ); // set my
client name (exe+process no)

        SERVERSTATUS sts;
        theSrv.GetStatus( out sts );

        // get infos about OPC server
        StringBuilder sb = new StringBuilder( sts.szVendorInfo, 200 );
        sb.AppendFormat( "    ver:{0}.{1}.{2}", sts.wMajorVersion,
sts.wMinorVersion, sts.wBuildNumber );
        txtServerInfo.Text = sb.ToString();

        // set status bar text to show server state
        sbpTimeStart.Text = DateTime.FromFileTime( sts.ftStartTime
).ToString();

        sbpStatus.Text = sts.eServerState.ToString();
    }
    catch( COMException e )
    {
        MessageBox.Show( this, "connect error!" + e.Message, "Exception",
MessageBoxButtons.OK, MessageBoxIcon.Warning );
        return false;
    }
    return true;
}
```


Group creation

```

public bool CreateGroup()
    {
        try
            {
                // add our only working group
                theGrp = theSrv.AddGroup( "OPCdotNET-Group", true, 500 );

                // add event handler for data changes
                theGrp.DataChanged += new DataChangeEventHandler(
this.theGrp_DataChange );
                theGrp.WriteCompleted += new WriteCompleteEventHandler(
this.theGrp_WriteComplete );
            }
        catch( COMException )
            {
                MessageBox.Show( this, "create group error!", "Exception",
MessageBoxButtons.OK, MessageBoxIcon.Warning );
                return false;
            }
        return true;
    }

```

Recursively call of OPC namespace Tree

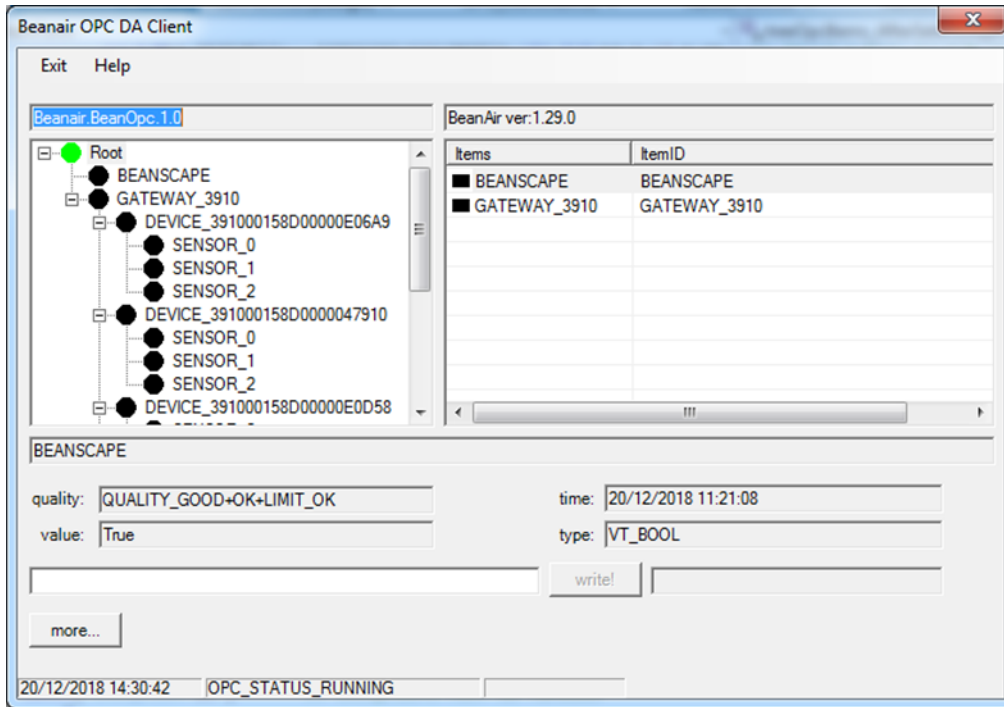
```

// recursively call the OPC namespace tree
public bool RecurBrowse( TreeNode tnParent, int depth )
    {
        try
            {
                ArrayList lst;
                theSrv.Browse( OPCBROWSETYPE.OPC_BRANCH, out lst );
                if( lst == null )
                    return true;
                if( lst.Count < 1 )
                    return true;

                foreach( string s in lst )
                    {
                        TreeNode tnNext = new TreeNode( s, 0, 1 );
                        theSrv.ChangeBrowsePosition(
OPCBROWSEDIRECTION.OPC_BROWSE_DOWN, s );
                        RecurBrowse( tnNext, depth + 1 );
                        theSrv.ChangeBrowsePosition(
OPCBROWSEDIRECTION.OPC_BROWSE_UP, "" );

                        tnParent.Nodes.Add( tnNext );
                    }
            }
        catch( COMException /* eX */ )
            {
                MessageBox.Show( this, "browse error!", "RecurBrowse",
MessageBoxButtons.OK, MessageBoxIcon.Warning );
                return false;
            }
        return true;
    }

```



SCIGATE AUTOMATION (S) PTE LTD

No.1 Bukit Batok Street 22 #01-01 Singapore 659592
 Tel: (65) 6561 0488 Fax: (65) 6562 0588
 Email: sales@scigate.com.sg Web: www.scigate.com.sg
 Business Hours: Monday - Friday 8.30am - 6.15pm